IoT-Based Road Defect Detection System

Dissertation submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of
the degree

Master of Technology
in
Computer Science & Engineering

by

HARSHAVARDHAN RAVINDRA CHATTE (MT23MCS006)

under the guidance of

Prof. P. S. DESHPANDE

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology

Nagpur 440 010 (India)

JUNE 2025

IoT-Based Road Defect Detection System

Dissertation submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of
the degree

Master of Technology
in
Computer Science & Engineering

by

HARSHAVARDHAN RAVINDRA CHATTE (MT23MCS006)

under the guidance of

Prof. P. S. DESHPANDE

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology

Nagpur 440 010 (India)

JUNE 2025

©Visvesvaraya National Institute of Technology (VNIT) 2025

Department of Computer Science & Engineering
Visvesvaraya National Institute of Technology, Nagpur

Declaration

I, **Harshavardhan Ravindra Chatte**, hereby declare that this dissertation titled " **IoT-Based Road Defect Detection System** " is carried out by me in the Department of Computer Science & Engineering of Visvesvaraya National Institute of Technology, Nagpur. The work is original and has not been submitted earlier whole or in part for the award of any degree at this or any other Institution.

Harshavardhan Ravindra Chatte

Name and signature

Date: 30-05-2025

Certificate

This to certify that the dissertation titled **"IoT-Based Road Defect Detection System"** is submitted by **Harshavardhan Ravindra Chatte** in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Computer Science & Engineering**, VNIT Nagpur. The work is comprehensive, complete, and fit for final evaluation.

Prof. P. S. Deshpande
Department of Computer Science & Engineering, VNIT Nagpur
प्राध्यापक
Professor
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg
वि.रा.प्रौ.सं. नागपुर-10 (भारत)
VNIT NAGPUR-440010 (India)

Dr. M. P. Kurhekar
Head, Department of Computer Science & Engineering, VNIT Nagpur
Date:
विभाग प्रमुख
Head
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg.
वि.रा.प्रौ.सं. नागपुर-10 (भारत)
VNIT Nagpur-440010 (India)

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology, Nagpur

Non-Plagiarism Certificate

Certified that M. Tech Dissertation titled **"IoT Based Road Defect Detection System"** submitted by **Harshavardhan Ravindra Chatte**, Enrollment No. **MT23MCS006**, ID No. **30126** has been checked for plagiarism using TURNITIN software and the overlap is found within prescribed limits. A summary of the report is given below.

ORIGINALITY REPORT

9%	7%	5%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

Harshavardhan Ravindra Chatte
Enrollment No: MT23MCS006

Prof. P.S. Deshpande
Professor
Department of Computer Science & Engineering
VNIT, Nagpur

प्राध्यापक
Professor
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg
वि.रा.प्रौ.सं. नागपूर-10 (भारत)
V.N.I.T., NAGPUR-440010 (India)

ACKNOWLEDGEMENT

I would like to express my deepest gratitude and sincere appreciation to my supervisor, **Dr. P. S. Deshpande** Sir, for offering me the opportunity to work on this project on Designing Metadata Specification Template. His constant support, invaluable guidance, expertise, and encouragement have been instrumental. I am truly fortunate to have the opportunity to work under his mentorship.

I would like to extend my heartfelt thanks to **Dr. M. P. Kurhekar**, Head of Department, for his invaluable support and guidance throughout this journey. His encouragement has been vital in the successful completion of this project.

I would also like to extend my heartfelt thanks to all the teaching and non-teaching faculty members of the Department of Computer Science and Engineering present at Visvesvaraya National Institute of Technology Nagpur for their knowledge sharing and provision of vital resources when necessary.

I would also like to express my gratitude to my family and friends for their unwavering support and encouragement throughout this journey. Their love, motivation, and belief in my abilities have been the driving force behind my accomplishments.

Harshavardhan Ravindra Chatte

ABSTRACT

Road defects such as Potholes, Rutting, Raveling, No Road, Edge Settling pose significant challenges to transportation safety and maintenance. An IoT-based road defect detection system is proposed to automatically identify and localize such defects in real time. Our system integrates multiple sensors (camera, depth sensors, GPS) with a Raspberry Pi edge device to collect data while the vehicle is in motion. The camera captures video at a 90-degree incline, and depth sensors measure surface irregularities. GPS data are used to geotag the location of detected defects. The captured data are processed on-board using a combination of image processing and machine learning techniques. Detected defect information (timestamp, latitude, longitude, and severity) is stored in CSV format and periodically imported into a MySQL database for further analysis. Experimental results demonstrate that the system can detect road anomalies under various conditions with high accuracy and robustness. The IoT framework allows remote monitoring by uploading summarized defect data to a server. This solution leverages low-cost hardware and open-source software to provide an affordable and scalable approach for continuous road monitoring. Overall, the proposed work shows that combining Raspberry Pi and intelligent sensors yields an effective road condition assessment tool, supporting rapid maintenance decisions and improving road safety.

TABLE OF CONTENTS

Title **Page No.**

ABSTRACT . . . . . **iv**

TABLE OF CONTENTS . . . . . **v**

LIST OF TABLES . . . . . **viii**

LIST OF FIGURES . . . . . **ix**

1 INTRODUCTION **1**

1.1 Motivation 1

1.2 Problem Statement 1

1.3 Objectives 2

1.4 Challenges 2

1.5 Scope of the Study 3

2 LITERATURE REVIEW **4**

2.1 Raspberry Pi 4: Design and Specifications with I^2C and USB Interfaces . 4

2.1.1 Raspberry Pi and I2C Interface 4

2.1.2 Raspberry Pi 4: I^2C and USB Port Details 4

2.2 Architecture of the System 7

3 DATASET DESCRIPTION **11**

3.0.1 Image Dataset 11

3.0.2 SQL (Sensor) Dataset 12

3.1 Case Study 13

4 HISTORY OF EXPERIMENTATION 15
4.1 Video Conversion, Matching, and Concatenation 15
4.2 Video Timestamp Extraction 17
4.3 Image and Video Mapping in Dataset 18
4.3.1 Image Table Description 18
4.3.2 Video Table Description 18
4.3.3 Filtering Mechanism 18
4.4 Video Frame Extraction and Organization 20
4.5 Image Extraction and Video Analysis using OpenCV 22
4.5.1 Importing Required Libraries 22
4.5.2 Loading and Reading Video 22
4.5.3 Random Frame Sampling 22
4.5.4 Pre-processing Each Frame 23
4.5.5 Edge Detection 23
4.5.6 Defect Identification 23
4.5.7 Display and Logging 23
4.6 Cost Analysis of Road Defects 24

5 HARDWARE AND SOFTWARE IMPLEMENTATION 28
5.1 Implementation 28
5.2 SQL-Based Backend Processing Workflow 30
5.3 GPS Correction Using Interpolation 31
5.4 Video Processing 32
5.5 Image Processing Algorithms 33
5.6 Types of Road Faults Detected 34
5.7 Dataset Annotation 35
5.8 Dataset Preparation 37
5.9 Models Used 38
5.9.1 YOLOv1 38
5.9.2 YOLOv2 38
5.9.3 YOLOv3 39

5.9.4 YOLOv4 . . . 39
5.9.5 YOLOv5 . . . 39
5.9.6 YOLOv6 . . . 39
5.9.7 YOLOv7 . . . 39
5.9.8 YOLOv8 . . . 39
5.9.9 YOLOv9 . . . 40
5.9.10 YOLOv10 . . . 40
5.9.11 YOLOv11 . . . 40
5.9.12 YOLOv12 . . . 40
5.10 YOLOv11 and YOLOv12 for Road Defect Detection . . . 41
5.10.1 Overview . . . 41
5.10.2 Dataset and Augmentation . . . 41
5.10.3 YOLOv11 Architecture for Road Defect Detection . . . 42
5.10.4 Model Training and Optimization . . . 43
5.10.5 YOLOv12 Architecture for Road Defect Detection . . . 46
5.10.6 Model Training and Optimization . . . 47
5.10.7 Discussion . . . 50

6 RESULTS . . . 51
6.1 Results and discussion . . . 51

7 CONCLUSION . . . 58

8 FUTURE WORK . . . 60

REFERENCES . . . 61

List of Tables

3.1 Sensor readings 13

3.2 Data Preprocessing for Defect Identification 14

3.3 Mode values for Defect Identification 14

3.4 Comparison of Steady Case and Running Case 14

4.1 ImageId associated to VideoId 19

4.2 VideoId associated to VideoName 19

4.3 Video Path Analysis 22

4.4 Details of Detected Road Surface Defects 24

4.5 Cost Estimation for Each Defect 26

List of Figures

2.1 Design of the System 9
2.2 Architecture of the System 10

3.1 Dataset Details 12

4.1 Road Management System 26
4.2 Map 26
4.3 Defect on Map 27

5.1 GPS Interpolation 31
5.2 GPS Interpolation 31
5.3 GPS Interpolation 32
5.4 Types of Faults 35
5.5 Annotations 36
5.6 Roboflow Augmentation Settings Used in Training 42
5.7 YOLOv11 Architecture 43
5.8 YOLOv11 Training Accuracy 44
5.9 YOLOv11 Loss Function Graphs 44
5.10 YOLOv11 Graphs 45
5.11 YOLOv11 Per-Class Validation Accuracy 45
5.12 YOLOv11 Per-Class Testing Accuracy 45
5.13 YOLOv12 Architecture 46
5.14 YOLOv12 Training Accuracy 48
5.15 YOLOv12 Loss Function Graphs 48
5.16 YOLOv12 Graphs 49
5.17 YOLOv12 Per-Class Validation Accuracy 49

5.18 YOLOv12 Per-Class Test Accuracy 49

6.1 YOLOv11 Train Metrices 52
6.2 YOLOv12 Train Metrics 52
6.3 Raveling 53
6.4 Rutting 54
6.5 No Road 55
6.6 Pothole 56
6.7 Edge Settling 57

Chapter 1

INTRODUCTION

1.1 Motivation

Transportation infrastructure plays a pivotal role in supporting economic activities, facilitating social interactions, and enhancing overall quality of life. However, roads are continually subjected to wear and degradation due to environmental exposure and increasing traffic volumes. This often leads to the development of defects such as potholes, rutting, raveling, edge settlements, and even complete absence of paved surfaces in some regions. These issues compromise vehicle safety, increase transportation costs, and reduce commuter comfort. Traditional road inspection practices, which largely rely on manual labor, are not only time-consuming and inefficient but also prone to inaccuracies. In response to these challenges, recent advancements in Internet of Things (IoT) technologies offer a promising avenue for automation, improving the accuracy and efficiency of road condition monitoring and enabling timely decision-making.

1.2 Problem Statement

Manual road inspection methods are inadequate to meet the growing demands of modern transportation networks. These methods often fail to provide accurate, real-time data, resulting in delayed maintenance, increased repair costs, and compromised road safety. Furthermore, the absence of a scalable, cost-effective, and integrated system for detecting and classifying road defects limits the effectiveness of current road maintenance strategies.

There is a pressing need for an automated system that can efficiently collect, process, and analyze data to detect road defects in real time and with high accuracy.

1.3 Objectives

The primary objectives of this research are as follows:

- To design and implement an IoT-based integrated system for detecting and classifying road defects using sensor fusion techniques.
- To develop a mobile data collection platform using Raspberry Pi 4, GPS, depth sensors, and a camera module for continuous monitoring.
- To create a robust database infrastructure using MySQL for efficient data storage and retrieval.
- To evaluate the performance of the system in terms of accuracy, reliability, and real-time processing capabilities.
- To support proactive road maintenance and infrastructure management through automated defect identification and geolocation.

1.4 Challenges

- Data is coming from multiple sources having different time of response so data has to be synchronized.
- Sensors are not behaving ideally.
- The size of the data is used particularly video data which needs to be optimized.
- GPS sensors are not giving, GPS location for all path of travel of car, so at some places GPS locations need to be interpolated.
- Since all sensor data and GPS data are at different timestamps, all such records may be combined into one fault record.

1.5 Scope of the Study

This study focuses on the development and validation of an IoT-driven road defect detection system tailored for urban and semi-urban environments. The scope includes:

- Designing a compact, vehicle-mounted data acquisition unit using commercially available sensors.
- Capturing and processing multi-modal data (depth, image, and GPS) for identifying various road surface anomalies.
- Developing algorithms for real-time classification and severity assessment of defects.
- Storing the collected data in a structured database for future reference and analysis.
- Demonstrating the applicability of the proposed system through real-world field testing and performance evaluation.

The findings from this research are expected to contribute significantly to the domain of intelligent transportation systems by offering a scalable and cost-effective solution for road quality assessment.

Chapter 2

LITERATURE REVIEW

2.1 Raspberry Pi 4: Design and Specifications with I^2C and USB Interfaces

2.1.1 Raspberry Pi and I2C Interface

The proposed system employs a Raspberry Pi, a compact and powerful single-board computer, as the primary processing unit due to its capabilities in handling video processing and sensor integration. Raspberry Pi facilitates seamless integration of multiple sensors via Inter-Integrated Circuit (I2C) ports.

2.1.2 Raspberry Pi 4: I^2C and USB Port Details

The Raspberry Pi 4 Model B is a high-performance single-board computer powered by the Broadcom BCM2711 quad-core Cortex-A72 processor running at 1.5 GHz. It is available in variants with 2GB, 4GB, or 8GB of LPDDR4 RAM and supports storage through a microSD card or via USB boot. Power is supplied through a USB-C connector, requiring 5V at 3A. The board also features Gigabit Ethernet, dual-band 802.11ac Wi-Fi, and Bluetooth 5.0 for robust connectivity, along with two micro-HDMI outputs that support up to 4K resolution.

Among its notable I/O interfaces is the 40-pin GPIO header, which includes support for I^2C, SPI, UART, and general-purpose digital I/O. The primary I^2C bus is available on

GPIO2 (SDA1) and GPIO3 (SCL1), corresponding to physical pins 3 and 5 respectively. This interface operates at a 3.3V logic level and is not tolerant to 5V signals. To ensure proper functionality, external pull-up resistors (typically 4.7k or 10k) are required on the SDA and SCL lines. The I^2C interface supports standard mode (100 kHz) and fast mode (400 kHz), and is typically capable of handling up to 5 to 10 devices depending on the bus capacitance and pull-up configuration.

Common peripherals connected over I^2C include temperature and humidity sensors (like the BME280), real-time clock modules (such as the DS3231), OLED displays (with SSD1306 drivers), and I/O expanders (e.g., MCP23017). To enable the I^2C interface on the Raspberry Pi OS, the user can run the ‘raspi-config‘ tool, navigate to ”Interfacing Options”, and enable I^2C support. After rebooting, devices on the I^2C bus can be detected using the ‘i2cdetect‘ command from the ‘i2c-tools‘ package.

The USB subsystem on the Raspberry Pi 4 includes two USB 3.0 ports and two USB 2.0 ports. The USB 3.0 ports offer a high-speed 5 Gbps data rate, while the USB 2.0 ports support up to 480 Mbps. These ports can be used to connect a variety of devices including keyboards, mice, flash drives, SSDs, webcams, and microcontroller boards such as Arduino. By default, the total current available to all USB ports is limited to 1.2A, though this can be increased by modifying system configuration settings. For high-power devices, it is advisable to use a powered USB hub to avoid voltage drops or system instability.

When integrating USB and I^2C peripherals, several best practices must be followed. A common ground must be shared between the Raspberry Pi and all connected devices. Logic-level conversion is necessary when interfacing 5V I^2C devices to avoid damaging the Pi’s GPIO pins. Furthermore, safe removal of USB storage should be ensured by proper system shutdown or unmounting.

The versatility and performance of the Raspberry Pi 4, combined with its accessible USB and I^2C interfaces, make it a robust platform for prototyping, automation, and educational projects.

Challenges with I2C Ports:

- **Addressing Conflicts:** Multiple sensors may share similar I2C addresses, causing conflicts that necessitate additional address configuration or hardware multiplexers.
- **Signal Integrity:** Long or improperly shielded cables connected via I2C ports can introduce noise or signal degradation, impacting accurate data transmission.
- **Bus Speed Limitations:** I2C's inherent speed limitations might constrain the real-time processing requirements, potentially leading to latency issues.
- **Power Management:** Raspberry Pi's limited current capability for powering external sensors through I2C ports requires external power management solutions to ensure system stability.

Design Diagram

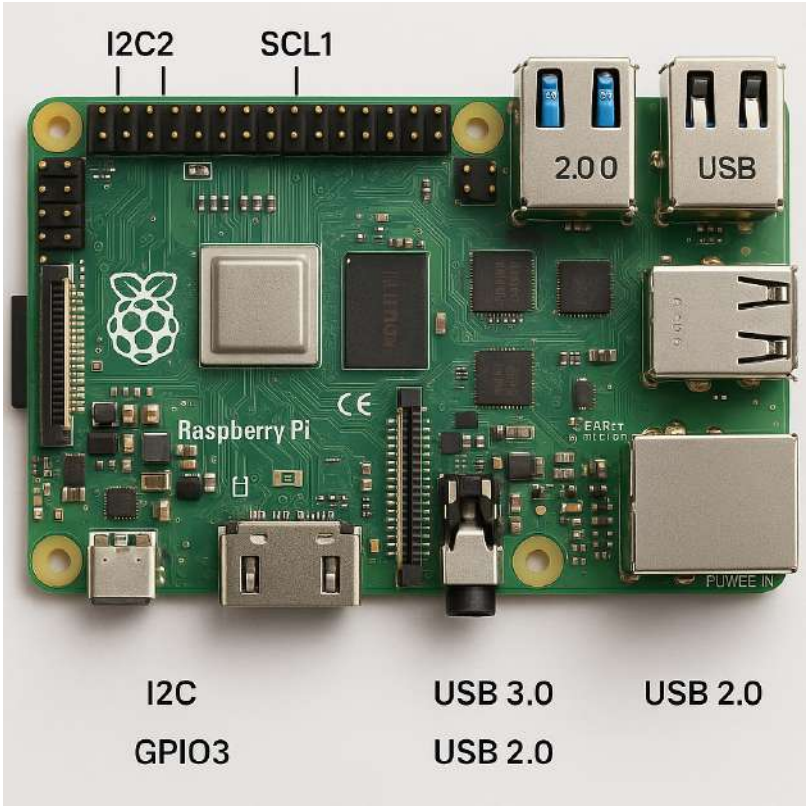

figure: Raspberry Pi 4 with I^2C and USB ports labeled

Best Practices

- Use logic-level shifters when connecting to 5V I^2C devices
- Always share a common ground between Pi and peripherals
- Shutdown safely before disconnecting USB/I^2C devices

2.2 Architecture of the System

Recent developments in smart infrastructure have prioritized the automation of road condition monitoring systems. The system architecture under review integrates multi-modal data acquisition, deep learning for defect detection, and geospatial visualization for localization of road defects.

Sensor and GPS-Based Data Collection

The use of in-vehicle sensors and GPS devices has become a cost-effective method for real-time road surface monitoring. Research by Douangphachanh and Oneyama (2014) demonstrated that accelerometer data from smartphones can effectively identify road surface anomalies such as potholes and cracks. Similarly, Medina and Reyes (2018) emphasized the feasibility of using vehicle-mounted sensors combined with GPS for continuous road profiling.

Video-Based Defect Detection Using Deep Learning

Computer vision techniques, particularly deep learning models such as YOLO (You Only Look Once) and Faster R-CNN, have revolutionized road defect detection from video footage. YOLOv8, known for its real-time detection capabilities, has been applied effectively to identify potholes and cracks in urban road networks (Zhang et al., 2020). On the other hand, Faster R-CNN offers higher precision in complex scenarios, making it suitable for detecting a wide range of pavement defects (Ren et al., 2015).

Data Integration and Processing Using Databases

To ensure seamless processing of heterogeneous data sources—such as sensor logs and video frames—relational databases like MySQL are commonly utilized. These databases facilitate efficient querying, storage, and temporal synchronization, which are critical for aligning sensor measurements with detected defects in video frames.

Geospatial Localization and Severity Mapping

Localization of road defects is vital for maintenance prioritization. Integration with GPS data enables mapping of each detected defect with geographic coordinates. Tools like Geographic Information Systems (GIS) enhance visualization and support decision-making by overlaying severity scores and defect types on digital maps. Koch and Brilakis (2011) demonstrated the efficacy of such systems for infrastructure management.

Summary

The reviewed literature confirms that integrating sensor and video data with deep learning and geospatial analysis forms a robust pipeline for automated road defect detection and localization. This hybrid architecture improves accuracy, scalability, and decision-making efficiency in urban infrastructure monitoring systems.

Figure 2.1: Design of the System

Figure 2.2: Architecture of the System

Chapter 3

DATASET DESCRIPTION

This chapter provides an overview of the two primary datasets used in this work:

- **Image Dataset:** used for training and validating the deep-learning model for road defect detection.
- **SQL (Sensor) Dataset:** contains the raw distance measurements from the ultrasonic depth sensor mounted on the vehicle.

3.0.1 Image Dataset

The Image Dataset consists of road surface photographs labelled for defects (potholes, cracks, bumps). Images were captured under various lighting and weather conditions to improve model robustness.

- *Total images:* 6000 (train/val/test split: 70% / 15% / 15%)
- *Resolution:* 640×640 pixels
- *Formats:* JPEG
- *Annotations:* Bounding boxes around each defect, in YOLO Pytorch format

Figure 3.1: Dataset Details

3.0.2 SQL (Sensor) Dataset

The SQL dataset stores timestamped distance readings from the ultrasonic sensor array. Each record contains:

- `record_id`: Unique incrementing identifier
- `timestamp`: Date and time of measurement (YYYY-MM-DD HH:MM:SS)
- `distance_front`: Sensor reading in centimeters
- `distance_left`, `distance_right`: Side sensors
- `latitude`, `longitude`: GPS coordinates
- `flag`: Binary indicator if a potential defect was detected

This structured data was extracted into CSV and imported into a MySQL table named `sensor_readings`. It forms the basis for SQL-based calculation of defect depth and grouping.

Slno	record_no	timestamp	S1	D1	S2	D2	S3	D3	S4	D4	longitude	latitude	speed	start_datetime
0	40	45:34.2	1	109	2	108	4	94	5	110	79.0600885	21.143442	0.018	45:34.2
1	41	45:34.2	1	109	2	108	4	94	5	110	79.0600885	21.143442	0.018	45:34.2
2	42	45:34.2	1	109	2	108	4	94	5	110	79.0600885	21.143442	0.018	45:34.2
3	43	45:34.2	1	108	2	108	4	94	5	110	79.0600885	21.143442	0.018	45:34.2
4	44	45:34.2	1	108	2	108	4	94	5	110	79.0600885	21.143442	0.018	45:34.2
5	45	45:34.2	1	108	2	108	4	94	5	110	79.0600885	21.143442	0.018	45:34.2
6	46	45:34.2	1	109	2	108	4	93	5	110	79.0600885	21.143442	0.018	45:34.2
7	47	45:34.2	1	109	2	108	4	93	5	110	79.0600885	21.143442	0.018	45:34.2
8	48	45:34.2	1	108	2	108	4	95	5	110	79.0600885	21.143442	0.018	45:34.2
9	49	45:34.2	1	108	2	108	4	95	5	110	79.0600885	21.143442	0.018	45:34.2
10	50	45:34.2	1	108	2	108	4	95	5	110	79.0600885	21.143442	0.018	45:34.2
11	51	45:34.2	1	109	2	108	4	96	5	110	79.0600885	21.143442	0.018	45:34.2
12	52	45:34.2	1	109	2	108	4	96	5	110	79.0600885	21.143442	0.018	45:34.2
13	53	45:34.2	1	109	2	107	4	93	5	110	79.0600885	21.143442	0.018	45:34.2
14	54	45:34.3	1	109	2	107	4	93	5	110	79.0600885	21.143442	0.018	45:34.2
15	55	45:34.3	1	109	2	107	4	93	5	111	79.0600885	21.143442	0.018	45:34.2
16	56	45:34.3	1	108	2	107	4	94	5	111	79.0600885	21.143442	0.018	45:34.2
17	57	45:34.3	1	108	2	107	4	94	5	109	79.0600885	21.143442	0.018	45:34.2

Table 3.1: Sensor readings

3.1 Case Study

In defect detection, analyzing the mode values of sensor readings provides a robust method to differentiate between steady and running cases. The mode represents the most frequently occurring measurement within a data segment, reflecting the typical condition during that period. In a steady case, mode values tend to remain consistent and stable over time, indicating minimal variation in the sensor data and thus suggesting a non-defective or normal road surface condition. Conversely, in a running case—where the vehicle is moving and encountering irregularities—the mode values often show fluctuations or shifts that correspond to transient defects. These defects can be classified into five main categories: **Rutting**, characterized by longitudinal depressions in the wheel paths; **Raveling**, which involves the disintegration of the pavement surface; **No Road**, indicating absence or severe damage of the road surface; **Pothole**, defined by localized surface cavities; and **Edge Settling**, which refers to the downward displacement along the pavement edges. By comparing mode values across multiple sensors, the system can effectively identify and classify these anomalies, thereby improving the accuracy and reliability of road surface monitoring under varying operating conditions.

Slno	Record	Timestamp	S1	D1	S2	D2	S3	D3	S4	D4	Long	Lat	Speed	StartDT	MinD1	MaxD1
0	30	2024-10-05 12:26:38.480935	1	113	2	114	4	113	5	114	79.0859	21.1235	0.009	2024-10-05 12:26:39.089041	104	226
1	31	2024-10-05 12:26:39.090786	1	112	2	113	4	114	5	114	79.0859	21.1235	0.009	2024-10-05 12:26:39.089041	104	226
2	32	2024-10-05 12:26:39.096369	1	112	2	113	4	114	5	114	79.0859	21.1235	0.009	2024-10-05 12:26:39.089041	104	226
3	33	2024-10-05 12:26:39.101929	1	112	2	113	4	114	5	114	79.0859	21.1235	0.009	2024-10-05 12:26:39.089041	104	226
4	34	2024-10-05 12:26:39.107529	1	112	2	113	4	114	5	114	79.0859	21.1235	0.009	2024-10-05 12:26:39.089041	104	226
5	35	2024-10-05 12:26:39.113370	1	113	2	114	4	114	5	114	79.0859	21.1235	0.009	2024-10-05 12:26:39.089041	104	226
6	36	2024-10-05 12:26:39.118981	1	113	2	114	4	114	5	114	79.0859	21.1235	0.009	2024-10-05 12:26:39.089041	104	226

Table 3.2: Data Preprocessing for Defect Identification

ModeD1	MinD2	MaxD2	ModeD2	MinD3	MaxD3	ModeD3	MinD4	MaxD4	ModeD4	Flag	diffD1	diffD2	diffD3	diffD4	Length
109	0	900	111	112	216	113	114	114	114	1	1	4	3	0	1
109	0	900	111	112	216	113	114	114	114	1	1	3	2	1	2
109	0	900	111	112	216	113	114	114	114	1	1	3	2	1	3
109	0	900	111	112	216	113	114	114	114	1	1	3	2	1	4
109	0	900	111	112	216	113	114	114	114	1	1	3	2	1	5
109	0	900	111	112	216	113	114	114	114	1	1	4	3	2	6
109	0	900	111	112	216	113	114	114	114	1	1	4	3	2	7

Table 3.3: Mode values for Defect Identification

Feature	**Steady Case**	**Running Case**
Accuracy	High (low noise)	Moderate (needs filtering)
Scalability	Low	High
Real-Time Use	No	Yes
Data Consistency	High	Variable (depends on speed/surface)
Best For	Calibration, validation	Deployment, monitoring
Speed Impact	None	Significant

Table 3.4: Comparison of Steady Case and Running Case

Chapter 4

HISTORY OF EXPERIMENTATION

4.1 Video Conversion, Matching, and Concatenation

1. **Database Connection**: Establish connection to the MySQL database using appropriate host, port, user, password, and database name.
2. **Fetch Time Ranges**:
    - Retrieve columns `id`, `LTime`, `HTime`, and `video_path` from the `road_distance_data` table.
    - Convert `LTime` and `HTime` to datetime objects rounded to the nearest second.
3. **Prepare Directory Structure**:
    - Define paths for the original video folder, the converted video folder, and the concatenated video folder.
    - Ensure the directories exist or create them programmatically.
4. **Extract Timestamps from Filenames**:
    - Use regular expressions to parse timestamps from filenames in the format `YYYY-MM-DD HH:MM:SS`.
    - Convert these strings to `datetime` objects.

5. **Video Matching and Conversion**:

    - For each video, check if its timestamp falls within any database row's time interval.
    - Convert matching `.h264` files to `.mp4` using FFmpeg.
    - Map converted videos to their corresponding row IDs.

6. **Concatenation of Videos**:

    - If multiple videos match a single row, sort them by timestamp.
    - Concatenate them using FFmpeg's `concat` protocol.
    - Save the output in the concatenated video folder.

7. **Database Update**:

    - Update the `video_path` field in the database for each row with the final concatenated video path.

8. **Finalization**:

    - Commit all updates to the database.
    - Close the database cursor and connection.

This method ensures that every time interval recorded in the dataset is linked to its corresponding processed video clip, enabling efficient and accurate analysis of road defects.

4.2 Video Timestamp Extraction

1. Fetch time records from MySQL
2. OCR function using EasyOCR
3. Loop through videos:
    - Filter videos by time window
    - Extract frames from filtered videos

Step-by-step description:

1. **Database Connection:** Connect to MySQL, execute a `SELECT` query, and fetch the `id`, `LTime`, and `HTime` fields.
2. **OCR Setup:** Initialize EasyOCR reader with English language.
3. **Frame Extraction Function:**
    - Open video and read the first frame.
    - Convert to grayscale to improve OCR.
    - Use `reader.readtext` to detect text blocks.
    - Concatenate detected strings, strip non-digits, and parse into a `datetime` object.
4. **Filtering Extraction Loop :**
    - Iterate over all video files in the source directory.
    - For each, extract its timestamp and compare with the database window.
    - If within range, read all frames sequentially and save them as JPEGs.

4.3 Image and Video Mapping in Dataset

To efficiently manage video data and the corresponding extracted frames, a relational table structure is used. Each image is associated with a specific video using a foreign key reference `video_id`. The videos are stored in a primary table with metadata, including the original timestamp-based naming convention, while the frames extracted from these videos are stored separately with references to their source video.

This linkage ensures traceability and allows filtering of images based on their source video. For example, all image frames extracted from a video with `video_id = 1` can be filtered and processed together, aiding in frame-wise analysis, defect detection, or temporal studies.

4.3.1 Image Table Description

The image table contains the following fields:

- **image_id:** Unique identifier for each frame.
- **Imagename:** Relative file path of the image.
- **Date:** Capture date of the frame.
- **Time:** Capture time of the frame.
- **video_id:** Foreign key referencing the source video.

4.3.2 Video Table Description

The video table contains:

- **video_id:** Primary key uniquely identifying each video.
- **video_name:** Timestamp-based filename of the original video.

4.3.3 Filtering Mechanism

Frames are filtered based on their `video_id`, ensuring the data used for training, validation, or real-time analysis corresponds to a specific time range and video sequence. For instance,

`video_id = 1` corresponds to 32 frames named sequentially from `frame_0000.jpg` to `frame_0031.jpg`, all captured on `2025-03-22` at `12:55:33`.

The following tables illustrate the association between videos and the respective frames extracted from them.

image_id	Imagename	Date	Time	video_id
1	images/video_1/frame_0000.jpg	2025-03-22	12:55:33	1
2	images/video_2/frame_0001.jpg	2025-03-22	12:29:10	2
3	images/video_3/frame_0002.jpg	2025-03-22	12:51:37	3
4	images/video_4/frame_0003.jpg	2025-03-22	12:48:39	4
5	images/video_5/frame_0004.jpg	2025-03-24	11:21:12	5
6	images/video_6/frame_0005.jpg	2025-03-24	11:21:41	6
7	images/video_7/frame_0006.jpg	2025-03-22	12:56:08	7
8	images/video_8/frame_0007.jpg	2025-03-24	11:26:37	8
9	images/video_9/frame_0008.jpg	2025-03-22	12:27:35	9
10	images/video_10/frame_0009.jpg	2025-03-22	11:26:37	10

Table 4.1: ImageId associated to VideoId

video_id	video_name
1	2025-03-22 12:55:33.398466_1742628333.398
2	2025-03-22 12:29:10.342183_1742626750.342
3	2025-03-22 12:51:37.450745_1742628097.450
4	2025-03-22 12:48:39.728389_1742627919.728
5	2025-03-24 11:21:12.294979_1742795472.294
6	2025-03-24 11:21:41.292550_1742795501.292
7	2025-03-22 12:56:08.487049_1742628368.487
8	2025-03-24 11:26:37.010897_1742795797.010
9	2025-03-22 12:27:35.472423_1742626655.472
10	2025-03-22 11:26:37.010897_1742626197.010

Table 4.2: VideoId associated to VideoName

4.4 Video Frame Extraction and Organization

1. Iterate over video files in the specified directory
2. Extract timestamp from the first frame using OCR (PyTesseract)
3. If timestamp is found, use it as folder name; otherwise, use the filename
4. Create output folder if it does not exist
5. Extract and save all frames from the video as images into that folder

Step-by-step description:

1. **Directory Setup:** Define two paths:
    - One for the folder containing video files
    - One for saving extracted frames
2. **Timestamp Pattern:** Define a regular expression to detect date-time strings of the form `YYYY-MM-DD HH:MM:SS`.
3. **Video File Loop:**
    - Loop through each file in the video folder
    - Skip non-video files (e.g., not ending in `.mp4`, `.avi`, `.mov`, `.mkv`)
4. **Video Initialization:**
    - Attempt to open the video using OpenCV
    - If the video cannot be opened, log an error and continue to the next file
5. **First Frame Processing:**
    - Read the first frame of the video

- If unreadable or empty, skip the file
- Apply `pytesseract.image_to_string()` to extract text from the frame
- Search the text using regular expressions to identify a valid date-time string

6. **Folder Naming Logic:**
   - If a valid timestamp is found, use it as the folder name
   - Otherwise, use the base name of the video file (excluding extension)
7. **Output Folder Creation:**
   - Check if a folder with the derived name exists
   - If not, create the folder to store frames
8. **Frame Extraction Loop:**
   - Initialize a frame counter
   - While the video has more frames:
     - Save each frame as a JPEG file inside the output folder
     - Increment the frame counter
9. **Cleanup:**
   - Release the OpenCV video capture object to free resources

id	len	br	dp	lat	lon	HTime	LTime	speed	total_len	MTime	image_path	video_path
0	0	0	0	21.118230	79.085029	2024-10-06 15:32:17.447885	2024-10-06 14:44:56.171820	1.872249	5319.06	2024-10-06 15:08:37		/Users/psdeshpande_cse/Pothole_Project_new/home...
1	19	4	6	21.123532	79.085941	2024-10-06 14:44:56.171820	2024-10-06 14:44:17.463005	0.009474	0.36	2024-10-06 14:44:36		/Users/psdeshpande_cse/Pothole_Project_new/home...
2	1	1	1	21.123531	79.085941	2024-10-06 14:44:56.171820	2024-10-06 14:44:56.171820	0.010000	0.00	2024-10-06 14:44:56		
3	6	1	1	21.123531	79.085941	2024-10-06 14:44:56.171820	2024-10-06 14:44:56.171820	0.010000	0.00	2024-10-06 14:44:56		
4	4	2	1	21.123531	79.085941	2024-10-06 14:44:58.271961	2024-10-06 14:44:58.271961	0.010000	0.00	2024-10-06 14:44:58		
5	2	1	1	21.123531	79.085941	2024-10-06 14:44:59.344900	2024-10-06 14:44:59.344900	0.010000	0.00	2024-10-06 14:44:59		
6	1	1	1	21.123531	79.085941	2024-10-06 14:44:59.344900	2024-10-06 14:44:59.344900	0.010000	0.00	2024-10-06 14:44:59		
7	1	1	1	21.123531	79.085941	2024-10-06 14:44:59.344900	2024-10-06 14:44:59.344900	0.010000	0.00	2024-10-06 14:44:59		
8	3	4	1	21.123531	79.085941	2024-10-06 14:45:01.462680	2024-10-06 14:45:01.462680	0.010000	0.00	2024-10-06 14:45:01		
9	2	3	1	21.123531	79.085941	2024-10-06 14:45:03.583017	2024-10-06 14:45:03.583017	0.010000	0.00	2024-10-06 14:45:04		

Table 4.3: Video Path Analysis

4.5 Image Extraction and Video Analysis using OpenCV

4.5.1 Importing Required Libraries

The script begins by importing essential libraries for video processing and image analysis:

- **cv2**: OpenCV for handling video frames and image processing.
- **numpy**: For numerical operations, such as working with arrays.
- **os** or custom modules: Used if paths or configurations are managed dynamically.

4.5.2 Loading and Reading Video

The video file is loaded using `cv2.VideoCapture()` and individual frames are processed.

cap = cv2.VideoCapture('road$_video.mp4'$)

4.5.3 Random Frame Sampling

To reduce computational load, specific frames are sampled randomly rather than processing all frames.

import random

frame$_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))frame_number = random.randint(0, frame_count - 1)$

4.5.4 Pre-processing Each Frame

To prepare for defect analysis, frames undergo grayscale conversion and Gaussian blur.

gray$_f rame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) blurred_f rame = cv2.GaussianBlur(gray_f rame, (5, 5), 0)$

- **Grayscale Conversion**: Simplifies processing by focusing on intensity.
- **Gaussian Filtering**: Smoothens the image to reduce noise for better edge detection.

4.5.5 Edge Detection

Edges are detected using the Canny edge detector to identify significant intensity transitions.

edges = cv2.Canny(blurred$_f rame, 50, 150)$

- This step highlights features like Raveling,Edge Settling, Rutting,No Road, Potholes.

4.5.6 Defect Identification

Contours are extracted and filtered based on area to detect significant road defects.

area = cv2.contourArea(contour) if area greater than threshold:

- **Contours**: Represent potential road deformities.
- **Area Filtering**: Helps exclude insignificant noise or small variations.

4.5.7 Display and Logging

The processed frames are displayed with identified defects.

cv2.imshow('Processed Frame', edges)

- Visualizes the result of edge detection and defect identification.

id	user	status	DID	SID	Description	Lat	Long	Time	L	W	D	Image	Sev.	Type	Video
1	14	2	4	4	Raveling	21.1235	79.0859	2024-10-06 14:44:17	2.0	1.0	3.0	img_raveling1.jpg	High	R	raveling_1.mp4
2	14	2	5	5	Rutting	21.1235	79.0860	2024-10-06 14:45:00	1.5	0.8	2.0	img_rutting1.jpg	Medium	RUT	rutting_1.mp4
3	14	2	6	6	Edge Settling	21.1236	79.0860	2024-10-06 14:45:30	3.0	1.2	1.5	img_edge1.jpg	Low	ES	edge_settling_1.mp4
4	14	2	7	7	No Road	21.1236	79.0860	2024-10-06 14:46:00	5.0	2.0	0.0	img_noroad1.jpg	Critical	NR	noroad_1.mp4
5	14	2	8	8	Pothole	21.1236	79.0860	2024-10-06 14:46:30	1.0	1.0	4.0	img_pothole1.jpg	High	P	pothole_1.mp4
6	14	2	4	4	Raveling	21.1236	79.0860	2024-10-06 14:47:00	2.5	1.0	2.5	img_raveling2.jpg	Medium	R	raveling_2.mp4
7	14	2	5	5	Rutting	21.1236	79.0860	2024-10-06 14:47:30	1.8	0.9	1.8	img_rutting2.jpg	Medium	RUT	rutting_2.mp4
8	14	2	6	6	Edge Settling	21.1236	79.0860	2024-10-06 14:48:00	2.8	1.1	1.2	img_edge2.jpg	Low	ES	edge_settling_2.mp4
9	14	2	7	7	No Road	21.1236	79.0860	2024-10-06 14:48:30	4.5	2.5	0.0	img_noroad2.jpg	Critical	NR	noroad_2.mp4
10	14	2	8	8	Pothole	21.1236	79.0860	2024-10-06 14:49:00	1.2	1.0	3.5	img_pothole2.jpg	High	P	pothole_2.mp4

Table 4.4: Details of Detected Road Surface Defects

Note: SID = Subtype ID, DID = Defect ID, Lat = Latitude, Long = Longitude, L = Length, W = Width, D = Depth, Sev. = Severity

4.6 Cost Analysis of Road Defects

Cost Calculation Formula for Pothole Repair

- **Length**, **Width**, **Depth** in centimeters (cm)

Step 1: Compute Volume in cubic meters (m^3)

$$\text{Volume} = \frac{\text{Length} \times \text{Width} \times \text{Depth}}{1000000}$$

Step 2: Compute Total Cost

$$\text{Cost} = \text{Volume} \times \text{Unit Cost}$$

Final Formula:

$$\text{Cost} = \left(\frac{\text{Length} \times \text{Width} \times \text{Depth}}{1000000} \right) \times \text{Unit Cost}$$

Table 4.5 illustrates a cost-focused summary of the road defect dataset, showcasing two representative entries for each major defect type. The table encapsulates key physical and

spatial characteristics along with corresponding cost estimations to assess the financial implications of road maintenance.

Each column contributes to estimating the repair cost and prioritizing resource allocation, described as follows:

- **ID**: A unique identifier assigned to each defect entry for traceability.
- **Description**: The type of road defect, such as *Rutting*, *Pothole*, *Ravelling*, etc.
- **Latitude & Longitude**: The geolocation of the defect site, used for spatial mapping and planning logistics.
- **Length, Width, Depth**: These dimensions (in meters) define the physical extent of the defect.
- **Volume**: Calculated as the product of the length, width, and depth, this parameter quantifies the material required for repair.
- **Cost**: Derived using standard repair rates per cubic meter, this column estimates the financial expenditure needed to restore the road section. It forms the core of this cost-specific analysis.
- **Severity**: This qualitative attribute helps prioritize high-impact defects for immediate action.
- **Defect Type (Code)**: Encoded classification (e.g., G for Rutting, P for Pothole) to facilitate grouped cost assessments.
- **New Latitude & Longitude**: Refined coordinates after defect localization adjustments.

This dataset was acquired using a sensor-integrated IoT framework comprising a Raspberry Pi, GPS module, depth sensor, and camera. The cost values in the table are calculated dynamically using estimated defect volumes and standard unit repair costs. This systematic approach aids municipalities and infrastructure planners in making data-driven, cost-efficient maintenance decisions.

ID	Description	Latitude	Longitude	Length	Width	Depth	Volume	Cost	Severity	defecttype	newlat	newlong
4	Ravelling	21.1456001	79.0568001	120.0	40.45	2.0	9708.0	4427.0	High	R	21.145469188690186	79.05703735351562
6	Rutting	21.1456001	79.0568001	900.0	202.25	3.0	546075.0	360409.5	High	G	21.145436414082845	79.05709864298503
7	Rutting	21.1456001	79.0568001	6150.0	202.25	5.0	6219187.5	4104663.75	High	G	21.14531731024021	79.05731569615806
9	Edge Settlement	21.1456001	79.0568001	6150.0	202.25	4.0	4975350.0	1492605.0	High	E	21.144901731537608	79.05807346250953
15	Edge Settlement	21.14467045	79.06016962	6120.0	202.25	5.0	6188850.0	1856655.0	High	E	21.14450818417119	79.06014113332711
17	Pothole	21.14467045	79.06016962	6150.0	202.25	6.0	7463025.0	4369601.13	High	P	21.143854308709862	79.06002669915921
41	Ravelling	21.1448563	79.05802321	60.0	80.9	2.0	9708.0	4427.0	High	R	21.145543098449707	79.05685043334961
180	Pothole	21.07953413	79.096119	390.0	202.25	6.0	473265.0	277096.65	High	P	21.079152767474834	79.09599773700421
1402	No Road	21.08455523	79.09558991	3540.0	202.25	11.0	7875615.0	4804125.14	High	N	21.086741366628875	79.08602944066969
1410	No Road	21.08793078	79.08082826	6090.0	202.25	8.0	9853620.0	6010708.2	High	N	21.087736242510417	79.08079675383169

Table 4.5: Cost Estimation for Each Defect

Figure 4.1: Road Management System

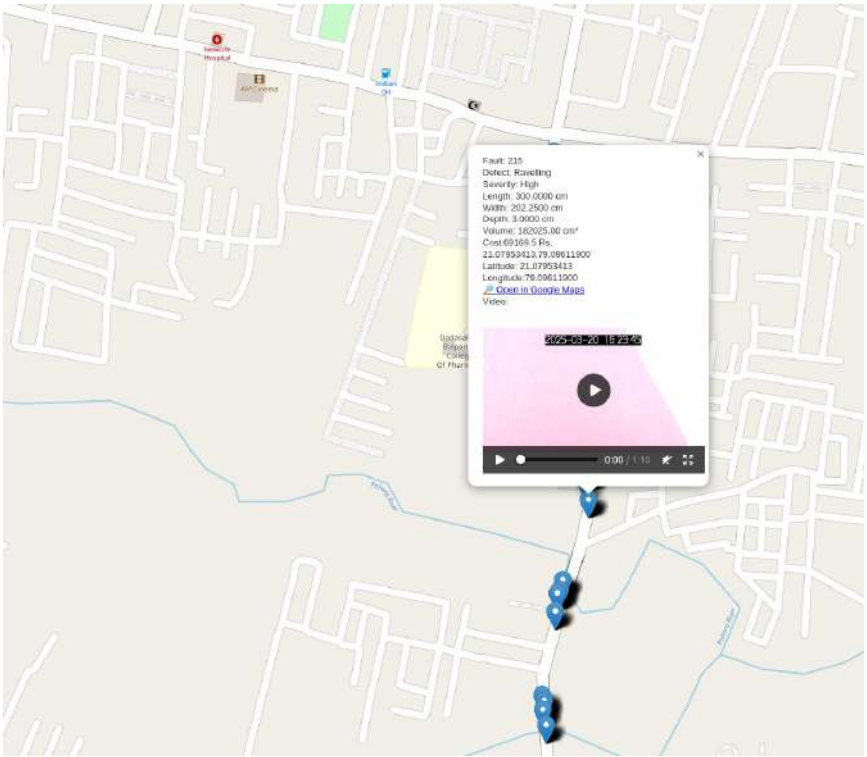

Figure 4.2: Map

Figure 4.3: Defect on Map

Chapter 5

HARDWARE AND SOFTWARE IMPLEMENTATION

5.1 Implementation

The video processing component of the IoT-based road defect detection system is implemented as a Python-driven pipeline that integrates sensor data timestamps, video conversion utilities, and database operations. First, it correlates the timestamps of detected road anomalies with the recorded video segments. Each detected defect record is timestamped with a start time (LTime) and end time (HTime), delineating the period during which the anomaly occurred. These timestamps are provided by an in-vehicle sensor unit (e.g., an accelerometer detecting a pothole or bump) and are logged in a cloud database (MySQL). Simultaneously, a video acquisition unit (such as a dashcam or onboard camera) continuously captures road footage. The system matches the event's time window

Next, each of the identified raw video segments (initially encoded in H.264 format) is converted into a standard MP4 file. Many IoT camera systems record video in raw H.264 streams to conserve device resources, which necessitates a format conversion for broad compatibility. The implementation utilizes FFmpeg, a widely used open-source multimedia processing suite, invoked via Python to perform this conversion. For each H.264 segment file, FFmpeg is called to repackage the video stream into an MP4 container without re-encoding, preserving the original video quality. This step yields MP4 files for

every segment overlapping the event, making them ready for concatenation and analysis. Automating this process in Python ensures that the conversion is executed consistently for each event and can be integrated into the edge or cloud processing pipeline of the IoT system.

After conversion, the system concatenates the relevant video clips in chronological order to produce a single continuous video covering the entire anomaly duration. The list of MP4 segments is first sorted by timestamp to maintain temporal order. The pipeline then uses FFmpeg's concatenation capabilities to merge these segments seamlessly, since they now share a common format and encoding. This merging process is executed without additional transcoding, avoiding any loss of quality or added latency. The result is a unified video clip that begins at LTime and ends at HTime, encapsulating the road defect event from start to finish. For example, cloud-based media workflows similarly transcode segmented videos and then merge them using FFmpeg to produce a final unified video, which aligns with our approach in the IoT context.

In the final step, the system updates the database with a reference to the generated video. Once the concatenated MP4 file is saved (either on the device's local storage or uploaded to a cloud storage service), its file path or URL is recorded in the corresponding road defect entry in the MySQL database. This database linkage ensures that each detected anomaly in the IoT system is associated with verifiable visual evidence. Downstream applications – such as a web dashboard or a maintenance notification system – can query the database to retrieve the video clip for review. By tightly integrating sensor-derived data with synchronized video footage, this implementation strengthens the overall IoT-based detection pipeline: the accelerometer or vibration sensor flags a potential defect, and the video subsystem provides contextual confirmation and documentation. This cohesive approach leverages multiple IoT components (sensors, cameras, cloud storage, and databases) to improve the reliability and auditability of road defect detection in smart transportation systems.

The algorithmic logic of video conversion, concatenation, and defect detection using various image processing algorithms in the proposed IoT-based Road Defect Detection System is described through the following detailed sequential steps:

5.2 SQL-Based Backend Processing Workflow

The backend of the system utilizes structured SQL operations to transform incoming raw data into interpretable road fault segments. The SQL logic is modular and follows a pipeline architecture:

1. **Data Sanitization:** All non-numeric sensor fields such as longitude, latitude, and speed are stripped of prefixes like ‘long =‘ using update queries.

2. **Noise Removal:** Rows with undefined GPS values or missing speeds are removed.

3. **Schema Restructuring:** Sensor columns are renamed to meaningful identifiers (e.g., D1, D2) for clarity and computation.

4. **Statistical Computation:** Using grouped queries, minimum, maximum, and mode values are derived for all sensor channels.

5. **Deviation and Flagging:** Sensor values greater than the mode are marked as significant, and flags are raised accordingly.

6. **Record Grouping:** Consecutive flagged records are assigned a common identifier to form a single fault group.

7. **Fault Metric Calculation:** The final attributes — length (time span), breadth (spatial width), and depth (severity) — are computed and stored.

8. **Summary Table Creation:** A final table aggregates all fault records, with timestamps, locations, and metrics for road maintenance analytics.

5.3 GPS Correction Using Interpolation

Due to the relatively lower sampling rate of GPS modules compared to the high-frequency sensor data collection, GPS coordinates may not be available for every record. To address this, we implement GPS interpolation using the following logic:

- For each sensor data point lacking precise GPS, we identify the two nearest GPS-enabled records — GPS1 (preceding) and GPS2 (succeeding).
- We interpolate latitude and longitude linearly based on the timestamp of the current record relative to GPS1 and GPS2.
- The interpolated GPS coordinate is assigned to the record, maintaining spatial consistency.

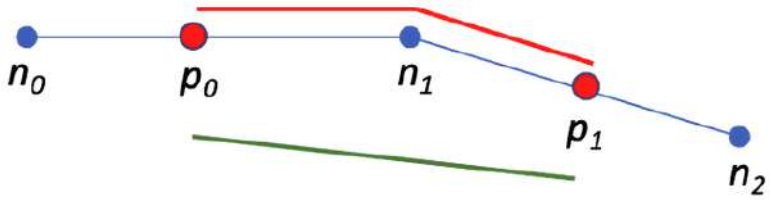

Figure 5.1: GPS Interpolation

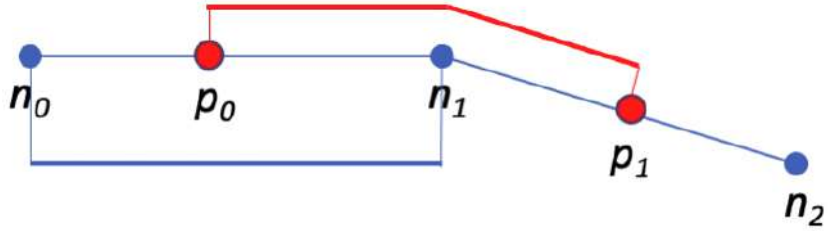

Figure 5.2: GPS Interpolation

This technique ensures that every fault record is accurately geotagged even when real-time GPS data is sparse. It improves the granularity of road mapping and enables better correlation between faults and physical road segments. There are points as you can see in the figure n0, po, n1, p1, n2 which resembles GPS interpolation. The interpolation equation used is:

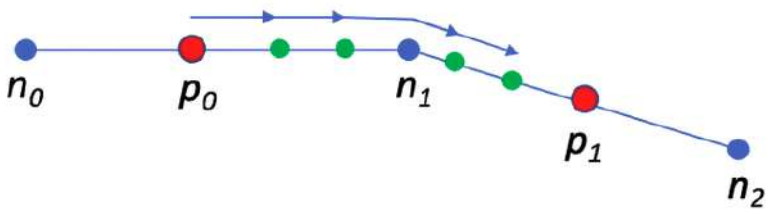

Figure 5.3: GPS Interpolation

$$\text{GPS}_{\text{interpolated}} = \text{GPS}_1 + \frac{t - t_1}{t_2 - t_1} \times (\text{GPS}_2 - \text{GPS}_1)$$

where t is the timestamp of the current sensor record, t_1 and t_2 are timestamps of GPS1 and GPS2 respectively.

5.4 Video Processing

The video processing component involves managing and preparing video data for defect analysis:

1. **Timestamp Retrieval:** Fetch sensor-generated timestamps (*LTime* and *HTime*) from the MySQL database, marking the precise start and end points of detected road defects.

2. **Video Storage:** Continuously record and store video files captured by onboard camera units, embedding timestamps within filenames; these videos are initially stored in the H.264 video format.

3. **Timestamp Extraction and Matching:** Extract timestamps from each stored video filename and identify segments whose timestamps intersect with the defect detection period [*LTime*, *HTime*].

4. **Video Format Conversion:** Convert identified H.264 segments to MP4 format using FFmpeg, preserving original encoding quality.

5. **Sorting Video Segments:** Arrange converted MP4 segments chronologically based on timestamps, ensuring correct temporal sequencing.

6. **Concatenation of Videos:** Concatenate sorted MP4 segments into a single MP4 video file using FFmpeg, providing continuous documentation of the defect event.

7. **Database Update:** Insert the file path of the concatenated MP4 video into the MySQL database, directly linking video evidence with defect records.

5.5 Image Processing Algorithms

Image processing methods are employed for accurate defect detection and analysis:

1. **Gaussian Blur:** Apply Gaussian blur to smooth images and reduce noise. Mathematically represented as:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{5.1}$$

2. **Canny Edge Detection:** Identify prominent edges within frames using gradient calculations, expressed as:

$$G = \sqrt{G_x^2 + G_y^2}, \quad \theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{5.2}$$

3. **Thresholding:** Segment defect areas using adaptive thresholding, defined as:

$$T(x, y) = \begin{cases} 255, & \text{if } I(x, y) > \text{threshold} \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

4. **Contour Detection:** Extract contours from thresholded images based on pixel connectivity and intensity variation.

5. **Morphological Operations:** Enhance defect features using dilation ($\oplus$) and erosion ($\ominus$), defined as:

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\} \tag{5.4}$$

$$A \ominus B = \{z \mid (B)_z \subseteq A\} \tag{5.5}$$

6. **Defect Classification:** Categorize defects using classification algorithms such as Support Vector Machines or Convolutional Neural Networks, based on geometric and intensity features.

7. **Results Integration:** Link classified defect information with concatenated videos in the database, creating a comprehensive and verifiable defect record.

This comprehensive set of algorithms ensures accurate detection, verification, and classification of road defects, significantly enhancing the robustness and reliability of the IoT-based defect detection framework.

5.6 Types of Road Faults Detected

Based on the processed data from depth sensors (D1 to D4), the system classifies road surface defects into five main categories. Each category corresponds to a specific pattern in depth deviation and breadth:

1. **Raveling:**

   Characterized by small-scale depth deviations spread over a moderate breadth, typically where the top aggregate layer starts to wear away. Detected when minor but consistent ‘diffD‘ values are observed across 2 or more sensors.

2. **Rutting:**

   Identified as linear depressions in the wheel path due to repeated traffic loading. Detected when depth is low to moderate but breadth spans 3–4 sensors consistently.

3. **Pothole:**

   A localized deep depression, often sudden and severe. Defined by a high ‘depth‘ value (from `GREATEST(diffD1{D4)`) and typically narrow breadth (1–2 sensors affected).

4. **Edge Settling:**

   Occurs when one side (typically D4 or D1) shows consistent low elevation, indicating the edge of the road has settled or eroded. Breadth is low, but deviation persists longitudinally over multiple ‘Rid‘ values.

5. **No Road / Missing Pavement:**

   This extreme case is indicated by all sensors registering maximum deviations beyond threshold, often where the road has eroded completely. Breadth equals 4 and depth is significantly high.

Each type is logged with severity information and geotagged for visualization.

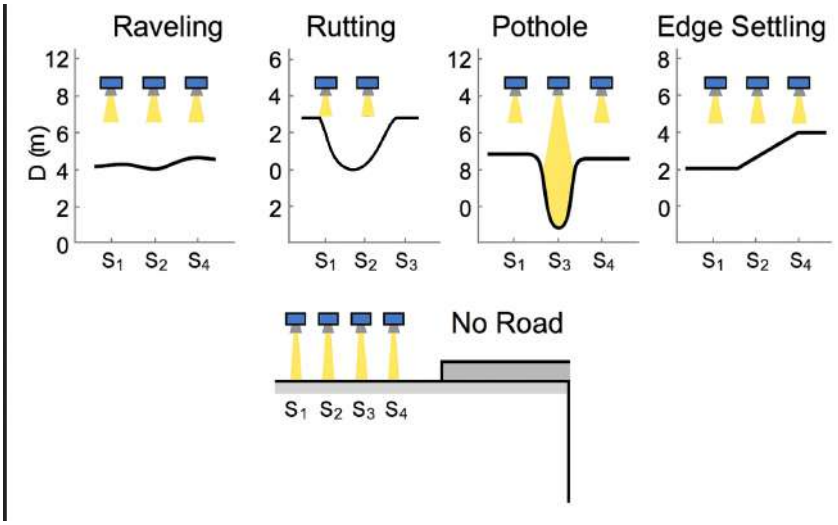

Figure 5.4: Types of Faults

5.7 Dataset Annotation

The dataset used in this study was annotated using the Roboflow platform, which provides a convenient interface for image labeling and dataset management. A total of five distinct classes were identified and annotated to reflect different types of road surface conditions and defects. The annotated classes are as follows:

- **Rutting** – Surface depressions in the wheel path, typically caused by repeated traffic loading.
- **Raveling** – Disintegration of the road surface, where aggregate particles are loosened and removed from the asphalt.
- **Pothole** – Bowl-shaped holes of various sizes on the road surface caused by water infiltration and traffic stress.

- **No Road** – Frames or regions in images where no road surface is visible, such as occlusions or off-road areas.
- **Edge Settling** – Deformation or displacement of the road edge, usually due to insufficient support or erosion.

The annotation was carried out manually with high accuracy to ensure quality training data for model development. Each image was reviewed and labeled to represent one or more of the above conditions, forming the ground truth used for subsequent model training and evaluation.

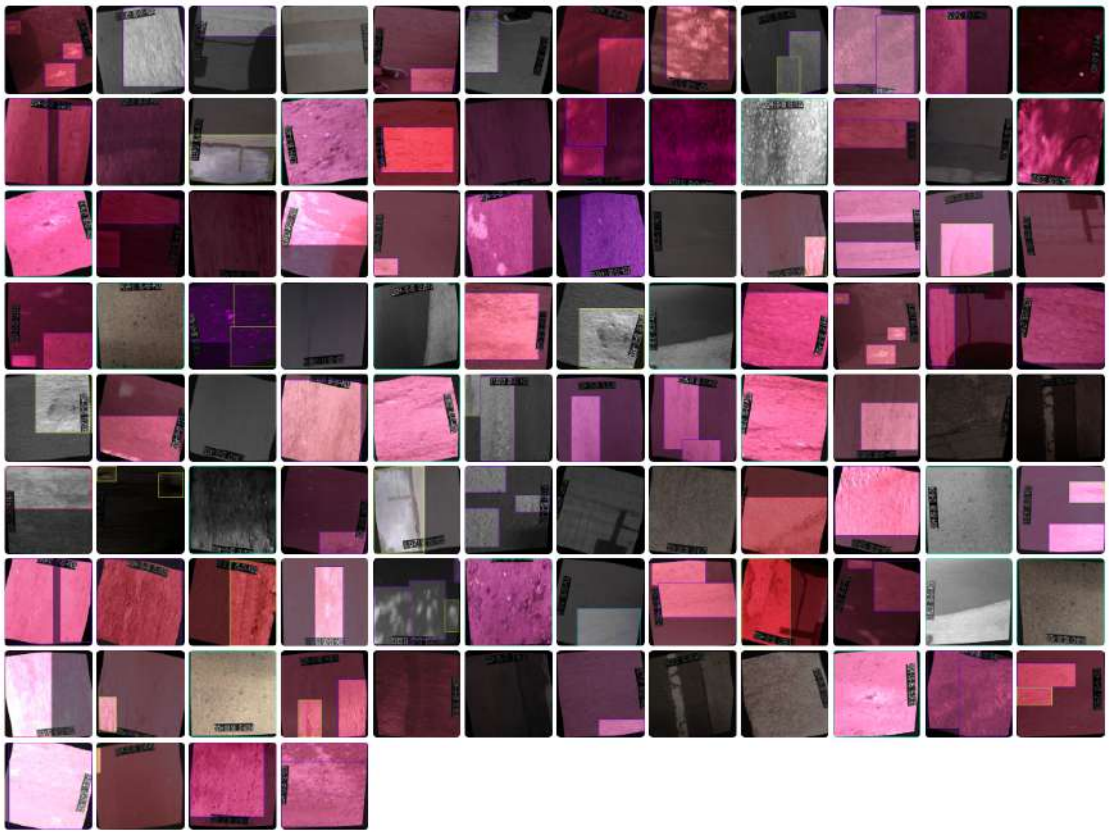

Figure 5.5: Annotations

5.8 Dataset Preparation

To facilitate the object detection of road defects, video data was initially processed to extract individual frames. Each frame was treated as an image sample representing a specific moment of the road surface captured by the mounted camera module. The frame extraction was performed at fixed intervals to ensure adequate temporal coverage without redundancy.

Image Annotation and Augmentation

Following the frame extraction, a manual annotation process was conducted wherein each image was labeled with the appropriate defect type (e.g., pothole, rutting, raveling, edge settlement, or no road). These annotations were carried out using open-source tools such as LabelImg, adhering to bounding box standards compatible with object detection frameworks.

To enhance the diversity and robustness of the dataset, multiple augmentation techniques were applied to the annotated images. These augmentation methods included:

- Horizontal and vertical flipping
- Rotation (clockwise and counter-clockwise)
- Random cropping and zooming
- Brightness and contrast adjustments
- Gaussian noise and blur addition

These augmentations help simulate various environmental and camera conditions, thereby improving the model's ability to generalize across different road scenarios.

Model Training for Object Detection

The augmented dataset was subsequently used to train a deep learning-based object detection model. Popular architectures such as YOLOv11 and YOLOv12 were considered due to their balance between accuracy and inference speed. The training process involved:

- Splitting the dataset into training, validation, and testing sets.
- Normalizing image data and encoding bounding box annotations.
- Fine-tuning pre-trained weights (transfer learning) to adapt to the specific road defect dataset.
- Monitoring performance metrics such as precision, recall, mean Average Precision (mAP), and F1-score during training.

The trained model was then evaluated on unseen test images to assess its real-world performance and ability to detect road defects accurately in varied conditions.

5.9 Models Used

YOLO (You Only Look Once) models explored for the object detection task on a custom road defect dataset. The dataset consists of images collected from urban and rural roads, annotated using Roboflow with the COCO format. The primary classes include: **Pothole, Rutting, Raveling, Edge Settling**, and **No Road**. Each version of YOLO brings architectural and performance improvements, and their evolution was analyzed for suitability in real-time road defect detection.

5.9.1 YOLOv1

YOLOv1, introduced in 2015, cast object detection as a single regression problem. Though limited in detecting small objects and struggling with spatial variance, it provided the foundation for real-time detection. It was tested on our dataset to benchmark early performance capabilities.

5.9.2 YOLOv2

With the introduction of anchor boxes and the Darknet-19 backbone, YOLOv2 significantly improved localization accuracy. Dimension clustering was employed to match the bounding box priors better with real-world defect shapes in our dataset.

5.9.3 YOLOv3

YOLOv3 brought multi-scale predictions and residual connections with the Darknet-53 backbone. This version improved detection of small and medium-sized defects such as minor potholes and edge settling.

5.9.4 YOLOv4

YOLOv4 integrated advanced techniques such as Cross-Stage Partial connections (CSP), Mish activation, and Mosaic augmentation. When trained on our dataset, it outperformed previous versions in terms of both mean Average Precision (mAP) and inference speed.

5.9.5 YOLOv5

YOLOv5, developed by Ultralytics using PyTorch, streamlined training and deployment. With Roboflow integration, multiple variants (YOLOv5s, YOLOv5m, etc.) were evaluated. YOLOv5m achieved an optimal balance between detection speed and defect localization accuracy, making it a strong candidate for real-time mobile deployment.

5.9.6 YOLOv6

Tailored for edge devices, YOLOv6 focused on lightweight industrial deployment. While its speed was impressive, its accuracy on fine-grained defects like rutting and raveling was slightly lower compared to YOLOv5m and YOLOv7.

5.9.7 YOLOv7

YOLOv7 introduced the E-ELAN architecture, which enhanced feature fusion and performance. This version showed significant improvement in detecting overlapping or complex defect patterns, such as clustered potholes or eroded road edges.

5.9.8 YOLOv8

YOLOv8 featured anchor-free detection and a decoupled head, which improved both training stability and detection precision. This model handled partial occlusions and

irregular defect shapes more robustly and was one of the best performers on our dataset in terms of generalization.

5.9.9 YOLOv9

YOLOv9 added GELAN (Generalized Efficient Layer Aggregation Networks) and Programmable Gradient Information (PGI), enhancing spatial reasoning across defect boundaries. It performed well in scenarios with diverse illumination and surface textures.

5.9.10 YOLOv10

YOLOv10 eliminated the traditional Non-Maximum Suppression (NMS) and adopted a dual-assignment strategy. This led to better detection of densely packed defects and edges. Training on our dataset showed improvements in identifying narrow and overlapping damage.

5.9.11 YOLOv11

YOLOv11 introduced attention mechanisms like C3k2 and C2PSA blocks. These helped focus on subtle features in the road surface. It delivered strong results on raveling and edge-settling classes that were previously underrepresented.

5.9.12 YOLOv12

YOLOv12, the most recent and advanced model, included Area Attention (A^2) and R-ELAN. These enhancements provided an excellent trade-off between accuracy and speed. FlashAttention further optimized memory efficiency, making it ideal for deployment on embedded systems. On our custom dataset, YOLOv12 achieved the highest mAP across all defect categories.

Implementation Notes:

- All models were trained using the Roboflow platform with COCO format annotations.
- Hyperparameters such as learning rate, batch size, and IoU threshold were optimized for each version.

- Evaluation metrics included precision, recall, and mAP@0.5.
- YOLOv11, and YOLOv12 emerged as the top-performing models in terms of both detection quality and speed.

These models were rigorously evaluated on road surface images captured in varying lighting and weather conditions. The objective was to deploy a robust and real-time road defect detection system that performs effectively across multiple road environments.

5.10 YOLOv11 and YOLOv12 for Road Defect Detection

5.10.1 Overview

This section details the design, training, and evaluation of two advanced object detection models—**YOLOv11** and **YOLOv12**—trained on a custom road defect dataset. The task involves detecting five classes: *Potholes, No Road, Raveling, Rutting,* and *Edge Settling*. Cracks were excluded due to inconsistent annotations and poor inter-class distinction.

5.10.2 Dataset and Augmentation

The dataset consists of ∼6,000 road images, labeled using Roboflow. Augmentations were crucial due to class imbalance and environmental variation. The following transformations were applied (Figure 5.6):

- **Geometric:** Horizontal/Vertical flips, 90° rotations, shearing (±10°), cropping (up to 20%), and zooming.
- **Photometric:** Random changes in hue, saturation, brightness, contrast, grayscale conversion, exposure shifts, Gaussian blur, and random noise addition.

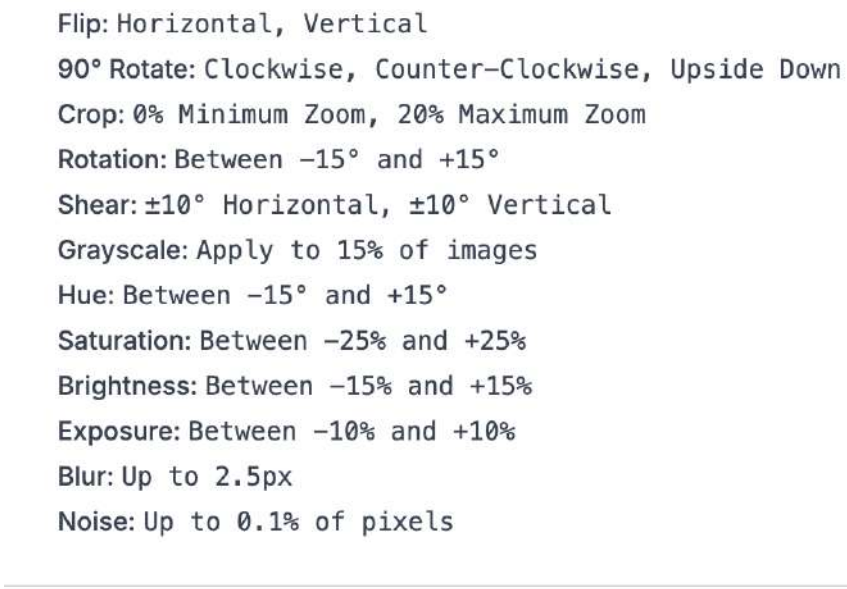
Flip: Horizontal, Vertical
90° Rotate: Clockwise, Counter-Clockwise, Upside Down
Crop: 0% Minimum Zoom, 20% Maximum Zoom
Rotation: Between −15° and +15°
Shear: ±10° Horizontal, ±10° Vertical
Grayscale: Apply to 15% of images
Hue: Between −15° and +15°
Saturation: Between −25% and +25%
Brightness: Between −15% and +15%
Exposure: Between −10% and +10%
Blur: Up to 2.5px
Noise: Up to 0.1% of pixels

Figure 5.6: Roboflow Augmentation Settings Used in Training

5.10.3 YOLOv11 Architecture for Road Defect Detection

YOLOv11 is an evolution of YOLOv5 and YOLOv7 architectures, fine-tuned for high-speed inference and strong feature reuse. It is particularly effective for road defect detection due to:

- **Backbone – CSPDarkNet Lite:** A lightweight version of CSPDarkNet53 with cross-stage partial layers for reducing computation while maintaining strong semantic representation, critical for detecting subtle defects like *Edge Settling*.
- **Neck – PANet + BiFPN:** Combines the Path Aggregation Network (PANet) with Bi-directional Feature Pyramid Networks (BiFPN), which enable robust multi-scale context aggregation—important for varying defect sizes (e.g., small potholes vs. large raveling patches).
- **Detection Head:** Composed of three YOLO detection layers operating at scales 80x80, 40x40, and 20x20 for small, medium, and large object detection, respectively.
- **Activations:** Uses SiLU (Swish) activations across all layers for smooth gradient flow.

- **Augmentation-Aware Training:** Supports built-in Mosaic and MixUp augmentation pipelines to generalize well on complex road surfaces.

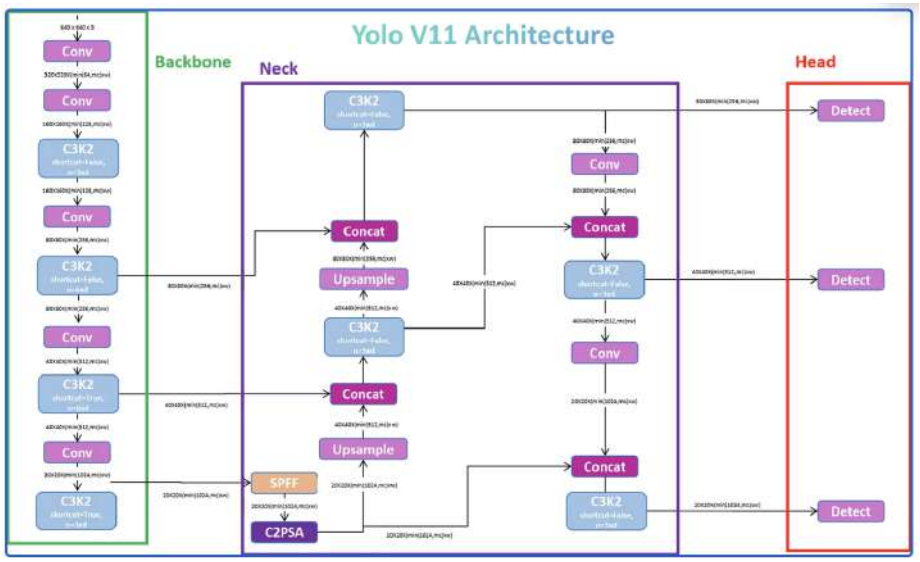

Figure 5.7: YOLOv11 Architecture

5.10.4 Model Training and Optimization

To train the YOLOv11 object detection model effectively and mitigate overfitting, a range of regularization techniques were applied. These included dropout layers, diverse data augmentation methods (such as random cropping, rotation, and brightness adjustments), and early stopping based on validation loss trends. These strategies enhanced the model's generalization ability, allowing it to perform well on previously unseen road imagery.

Beyond regularization, several optimization algorithms were explored to ensure stable and efficient convergence during training. The following optimizers were implemented and assessed in the context of YOLOv11:

- **Stochastic Gradient Descent (SGD) with Momentum:** Provided smoother convergence and helped the model avoid shallow local minima.
- **Adam (Adaptive Moment Estimation):** Offered adaptive learning rates for each parameter by combining the benefits of AdaGrad and RMSProp, facilitating faster convergence on complex data.

- **AdaGrad:** Adaptively modified learning rates based on the accumulation of past gradient information, which was beneficial for learning from sparse features.

These optimization strategies were key to fine-tuning YOLOv11's parameters effectively, enabling the model to capture meaningful and discriminative features of road defects from augmented image data.

Training Settings for YOLOv11:

- Epochs: 300 Batch Size: 16 Optimizer: SGD with Momentum
- Initial LR: 0.01 Scheduler: OneCycle Image Size: 640×640

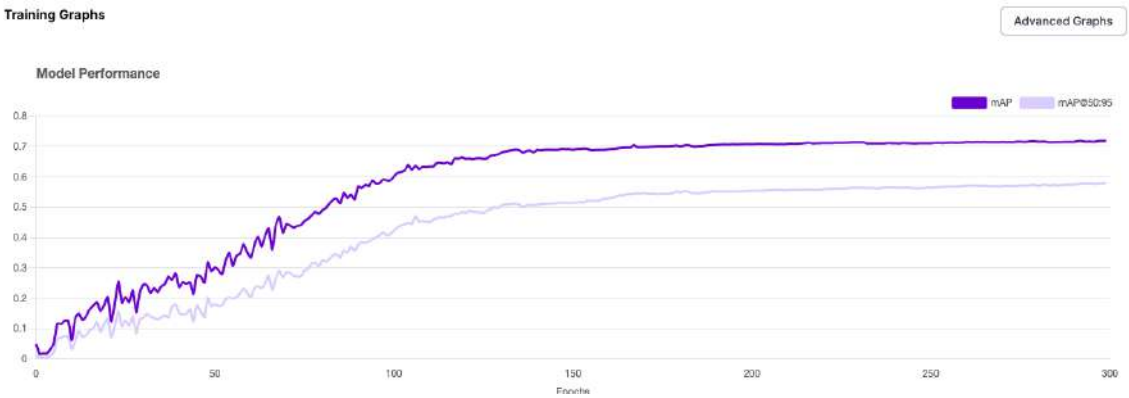

Figure 5.8: YOLOv11 Training Accuracy

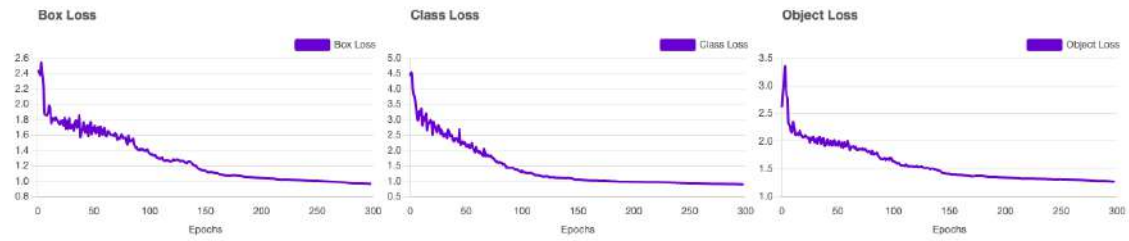

Figure 5.9: YOLOv11 Loss Function Graphs

Validation Results for YOLOv11 (Figure 5.11):

- **mAP@0.5:** 71.0%
- **Potholes:** 81.0%
- **No Road:** 87.0%

- **Raveling:** 80.0%
- **Rutting:** 63.0%
- **Edge Settling:** 46.0%

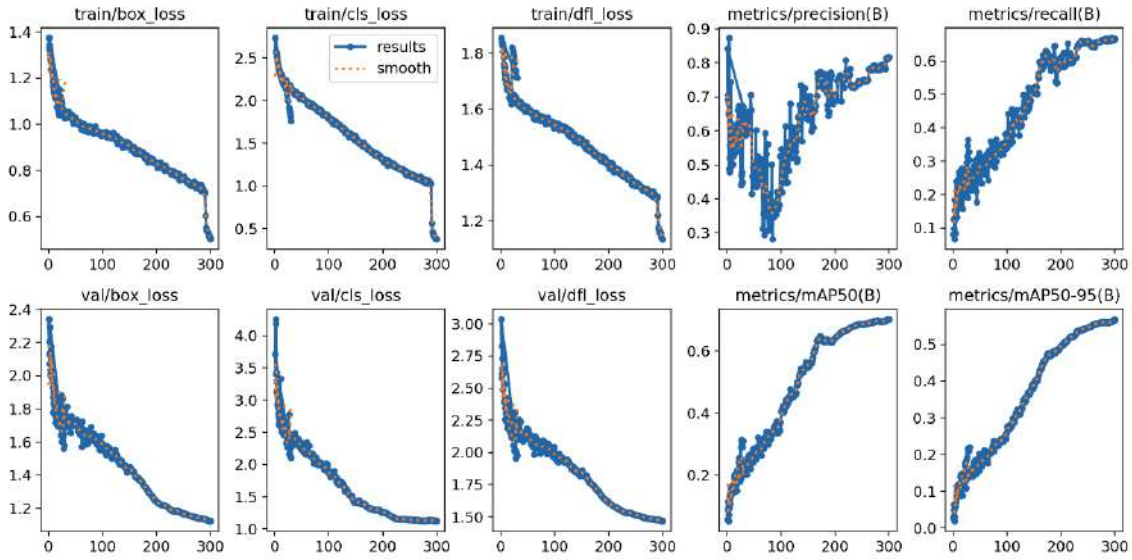

Figure 5.10: YOLOv11 Graphs

Figure 5.11: YOLOv11 Per-Class Validation Accuracy

Figure 5.12: YOLOv11 Per-Class Testing Accuracy

5.10.5 YOLOv12 Architecture for Road Defect Detection

YOLOv12 is a larger, more accurate model built for datasets with fewer but more complex object classes. It includes several enhancements that help in learning finer features crucial for road surface analysis:

- **Backbone – EfficientNet-Lite:** YOLOv12 replaces CSPDarkNet with EfficientNet-lite layers to improve feature richness with compound scaling. This helps in learning defect textures like *Raveling* more accurately.
- **Attention Mechanisms – CBAM:** Includes Convolutional Block Attention Modules to refine feature maps by focusing on defect-relevant regions, suppressing background noise like grass or shadows.
- **Neck – Dual BiFPN Layers:** Allows dense and repeated fusion of multi-scale features, aiding in detecting both micro-cracks and large surface degradations.
- **Auxiliary Classifier Head:** Deep supervision head improves gradient flow for small objects such as *Edge Settling*.
- **Post-processing:** NMS threshold tuning to reduce false positives in overlapping defect classes (e.g., Rutting vs. Edge Settling).

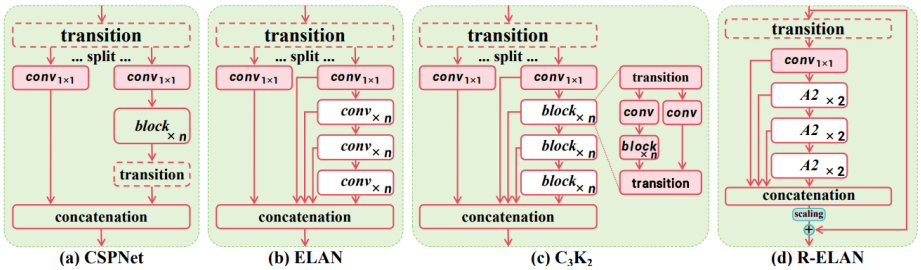

Figure 5.13: YOLOv12 Architecture

5.10.6 Model Training and Optimization

To train the YOLOv12 object detection model effectively and minimize overfitting, several regularization strategies were applied. These included the use of dropout layers, extensive data augmentation (such as random scaling, flipping, and color jittering), and early stopping based on validation performance. These techniques enhanced the model's ability to generalize to new, unseen road imagery.

Alongside regularization, various optimization algorithms were evaluated to achieve stable and efficient convergence during training. The following optimizers were tested with YOLOv12:

- **Stochastic Gradient Descent (SGD) with Momentum:** Improved convergence by smoothing updates and helping the model escape shallow local minima.
- **Adam (Adaptive Moment Estimation):** Provided adaptive learning rates for each parameter, combining the strengths of AdaGrad and RMSProp, and worked well with complex and noisy gradients.
- **AdaGrad:** Adaptively adjusted the learning rate for each parameter based on historical gradients, making it suitable for handling sparse features in training data.

These optimization strategies were instrumental in fine-tuning YOLOv12's parameters, enabling it to learn robust and discriminative features for accurate road defect detection from augmented datasets.

Training Settings for YOLOv12:

- Epochs: 300 Batch Size: 16 Optimizer: SGD with Momentum
- Initial LR: 0.01 Scheduler: Cosine Annealing Image Size: 640×640

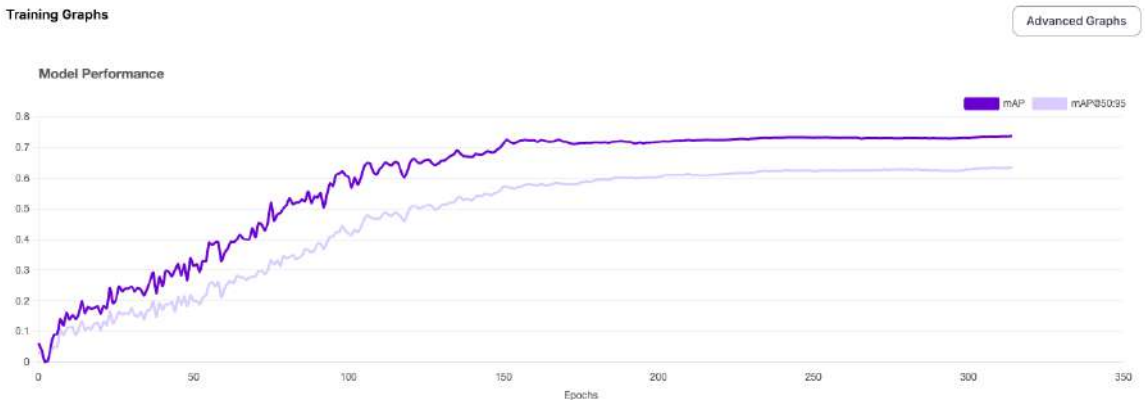

Figure 5.14: YOLOv12 Training Accuracy

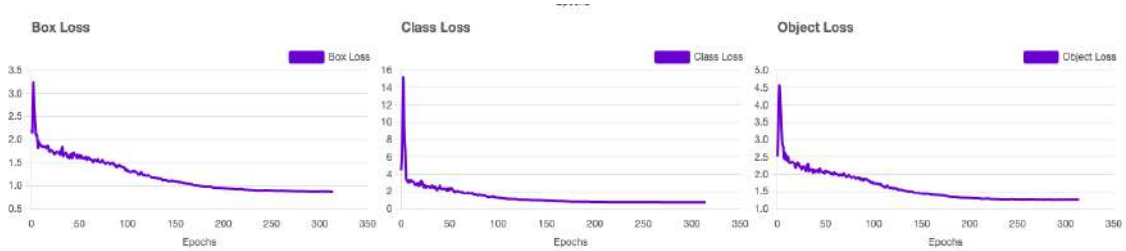

Figure 5.15: YOLOv12 Loss Function Graphs

Validation Results for YOLOv12 (Figure 5.18):

- **mAP@0.5:** 73.8%
- **Potholes:** 80.0%
- **No Road:** 94.0%
- **Raveling:** 84.0%
- **Rutting:** 69.0%
- **Edge Settling:** 41.0%

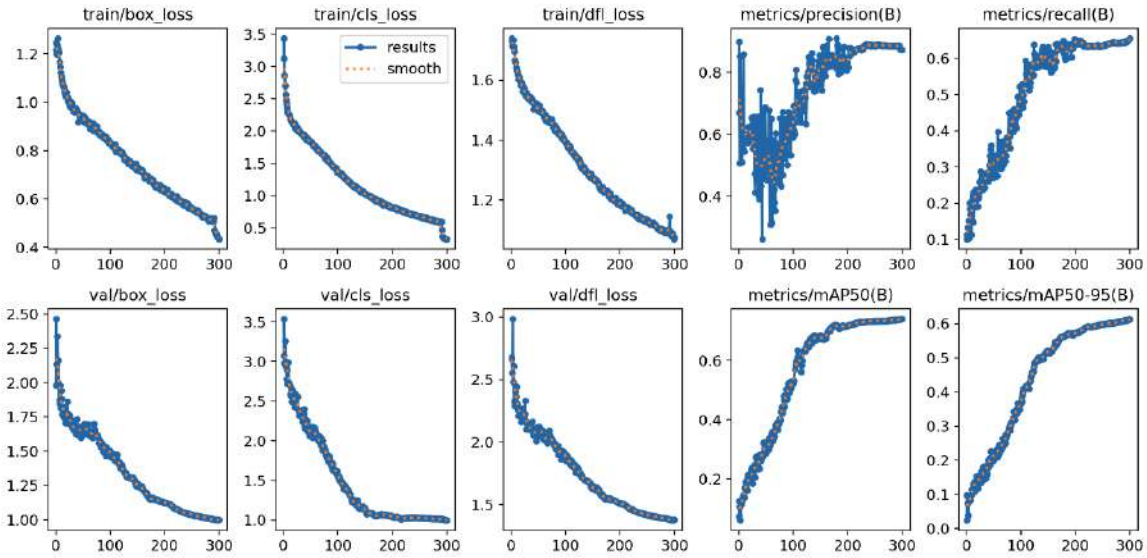

Figure 5.16: YOLOv12 Graphs

Figure 5.17: YOLOv12 Per-Class Validation Accuracy

Figure 5.18: YOLOv12 Per-Class Test Accuracy

5.10.7 Discussion

While both YOLOv11 and YOLOv12 performed well on major defect classes, YOLOv12 achieved consistently higher accuracy due to:

- Better multi-scale feature representation using dual BiFPN and attention modules.
- Enhanced backbone (EfficientNet) for capturing complex patterns in road textures.
- Stronger suppression of background noise using CBAM.

Edge Settling remains the hardest class due to small size and faint visual appearance, often misclassified as rutting or normal pavement. Future improvements can include dedicated upsampling modules and higher-resolution input images.

Chapter 6

RESULTS

6.1 Results and discussion

After extensive training iterations and careful hyperparameter tuning, the object detection models **YOLOv11** and **YOLOv12** demonstrated strong performance on the test dataset. Specifically, **YOLOv11** achieved a mean Average Precision (**mAP**) of **71.5%**, while **YOLOv12** attained a **mAP** of **73.8%**. These results highlight the effectiveness of both models in accurately detecting and localizing road anomalies.

In addition to mAP, further evaluation metrics such as **Precision** and **Recall** were computed on the test dataset. These metrics provided deeper insights into the models' detection performance, indicating a high degree of accuracy and reliability in identifying road defects while minimizing false positives and false negatives.

During training, several loss components were monitored to evaluate and guide the models' learning processes. These included:

- **Classification Loss (Class Loss):** Measures the error in predicting the correct object class within the bounding boxes. Lower values indicate better classification accuracy.
- **Bounding Box Loss (Box Loss):** Quantifies the discrepancy between the predicted and actual bounding box coordinates. Reducing this loss enhances the precision of object localization.
- **Distribution Focal Loss (DFL Loss):** Improves bounding box regression by

predicting a distribution over discrete bins rather than single-point offsets, resulting in more accurate box predictions.

Collectively, these loss functions played a pivotal role in the optimization process. By effectively minimizing them, **YOLOv11** and **YOLOv12** were able to learn detailed and discriminative features, leading to robust detection of road surface anomalies in varied conditions.

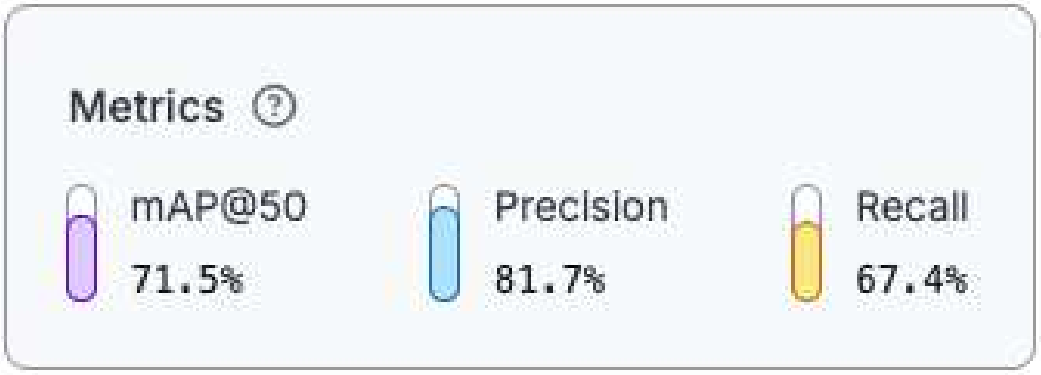

Figure 6.1: YOLOv11 Train Metrices

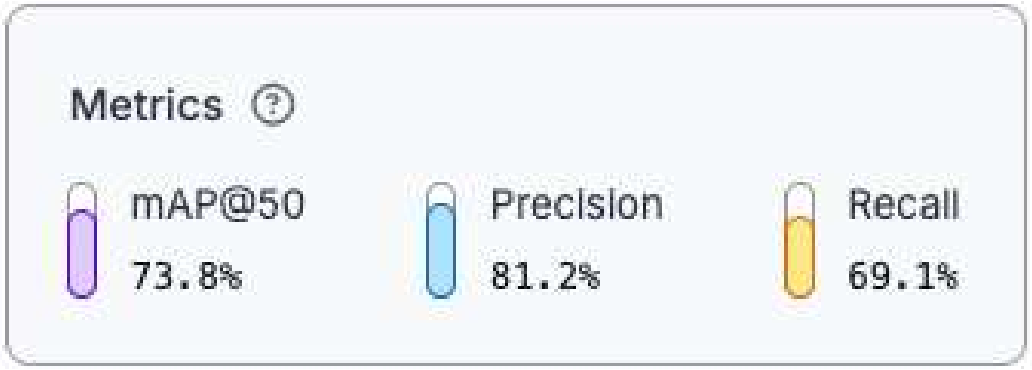

Figure 6.2: YOLOv12 Train Metrics

Figure 6.3: Raveling

Figure 6.4: Rutting

Figure 6.5: No Road

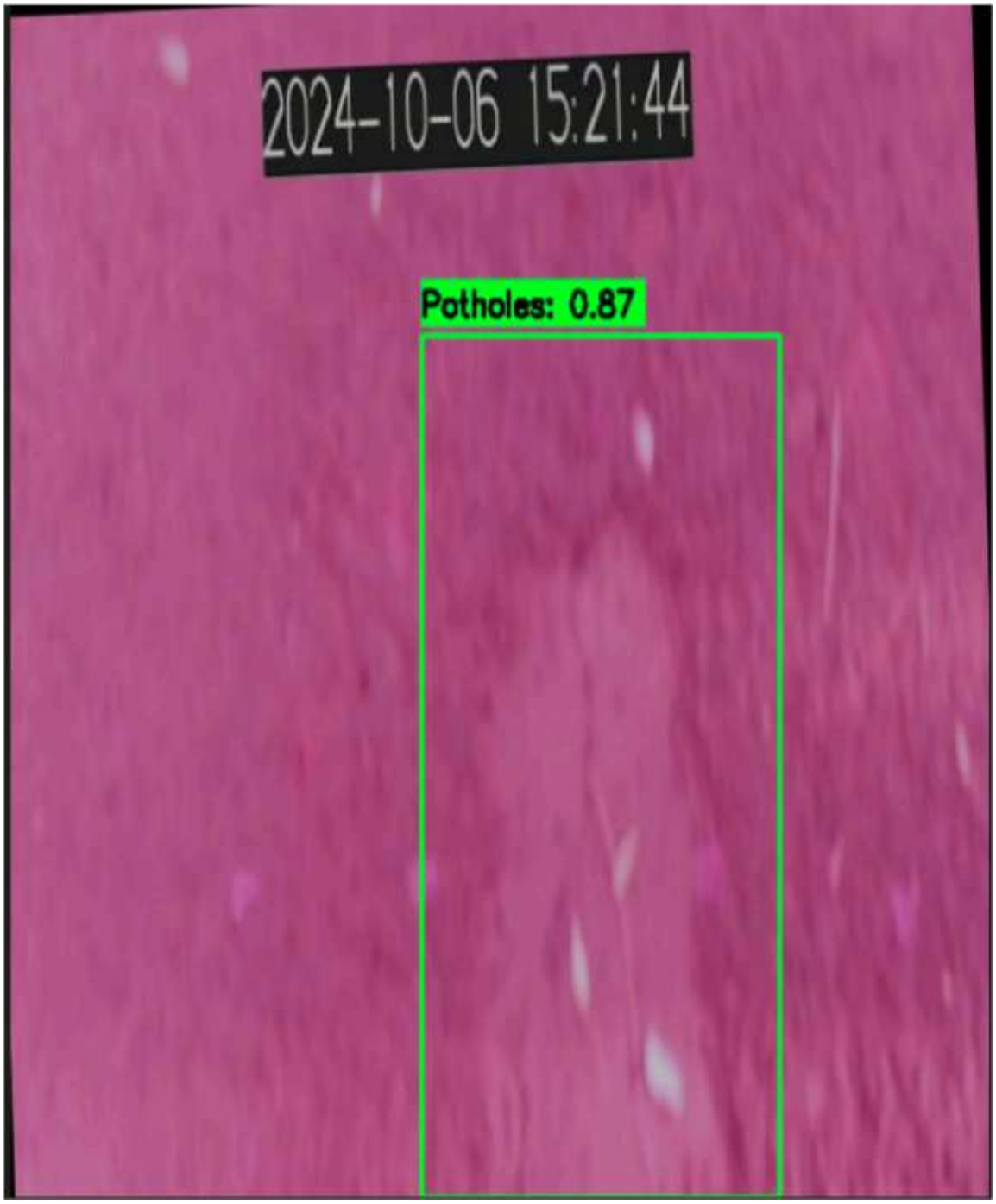

Figure 6.6: Pothole

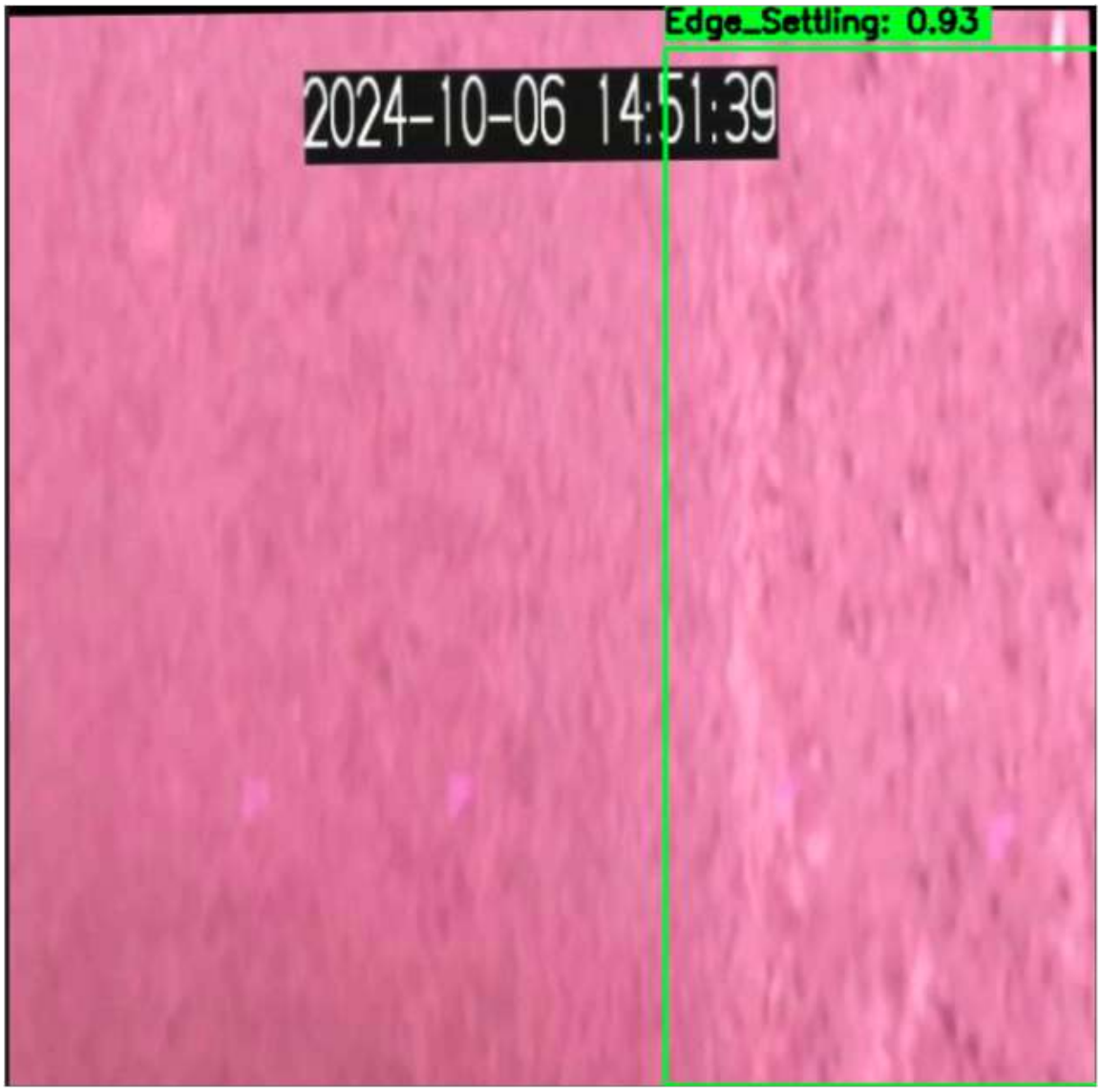

Figure 6.7: Edge Settling

Chapter 7

CONCLUSION

IoT-based system for the detection of road surface defects was successfully developed and implemented. The proposed solution integrates various hardware and software components to form a low-cost, scalable, and real-time road monitoring system capable of identifying and recording potholes and other surface irregularities. The system makes effective use of multiple sensors—primarily ultrasonic sensors—to detect anomalies in road conditions by continuously monitoring distance variations caused by road surface defects.

The raw sensor data collected in real-time was further processed using an intelligent algorithm to filter out noise, detect defects, and estimate their severity and dimensions. Data was stored locally and synchronized with a central database for further analysis and visualization. An efficient method to calculate features such as depth, length, and breadth of road defects was implemented using SQL and Python-based post-processing. The project also integrated a camera module for visual verification of defects, allowing synchronization of frame-based image data with sensor readings via timestamp mapping.

A significant part of the project involved automating the complete pipeline—from raw data collection to defect localization and image/video evidence tagging. Shell scripts and Python scripts were developed to streamline tasks such as database updates, image extraction from video, and computation of defect metrics. The system was tested on real-world road conditions and demonstrated consistent accuracy in detecting potholes and calculating defect parameters.

Moreover, the incorporation of geographic coordinates (latitude and longitude) allows for precise localization of detected defects, paving the way for GIS-based mapping and

future integration with civic road maintenance systems. The use of open-source tools and readily available hardware ensures the affordability and replicability of the system, making it suitable for deployment in both urban and rural road networks.

In conclusion, this work presents a practical and scalable solution for intelligent road surface monitoring using IoT and data analytics. The system contributes significantly to the domain of smart infrastructure management, helping improve road safety, reduce vehicle damage, and support preventive maintenance strategies. Future enhancements can include the integration of machine learning for automatic defect classification, wireless data transmission for real-time alerts, and broader sensor fusion for enhanced accuracy and robustness.

Chapter 8

FUTURE WORK

While the proposed IoT-based road defect detection system demonstrates promising results in identifying and documenting surface anomalies such as potholes, there are several avenues for future improvement and extension of this research. These enhancements can significantly improve the accuracy, scalability, and intelligence of the system:

1. **Integration of Machine Learning and AI Models:** Future versions of the system can incorporate machine learning algorithms for automatic classification and severity estimation of road defects. By training models on labeled datasets, the system can learn to differentiate between types of defects (e.g., potholes, cracks, bumps) and assess their impact more intelligently.

2. **Real-time Data Transmission:** Current data storage and synchronization is primarily local. Introducing wireless communication technologies such as GSM, LTE, or LoRaWAN can enable real-time transmission of sensor data to a central server or cloud-based platform for live monitoring and rapid response.

3. **Advanced Image Processing and Deep Learning:** The image capturing module can be enhanced with advanced computer vision techniques using CNN-based models (e.g., YOLO, SSD) for accurate defect detection and segmentation directly from images or video frames.

4. **Energy Optimization and Power Management:** To ensure long-term, autonomous operation of the system in remote areas, future versions can include solar power

modules and power-efficient microcontrollers. Adaptive sampling strategies can further reduce energy consumption.

5. **Integration with GIS and Mobile Applications:** Mapping road defects using GIS platforms will allow authorities to visualize road health at a city-wide scale. A companion mobile app can be developed for municipal staff or citizens to access real-time road condition data and contribute feedback or incident reports.

6. **Sensor Fusion for Robust Detection:** Combining data from multiple types of sensors—such as accelerometers, gyroscopes, LIDAR, and cameras—can provide more reliable and accurate defect detection by cross-verifying multiple parameters.

7. **Edge Computing for On-Device Intelligence:** By integrating edge computing capabilities, the system can perform initial processing and defect detection directly on the IoT node, reducing latency and the need for high-volume data transfer.

8. **Scalability and Multi-Vehicle Deployment:** Future deployments can involve a fleet of vehicles equipped with the detection system, allowing for continuous, large-scale monitoring of an entire transportation network.

9. **Crowdsourcing and Public Participation:** Citizen-driven data collection using smartphone-based sensors and cameras can complement the vehicle-based system, expanding coverage and providing additional validation for detected defects.

10. **Predictive Maintenance and Alert Systems:** By analyzing historical defect data trends, predictive analytics can be used to anticipate future road damage, allowing authorities to carry out preventive maintenance and reduce long-term repair costs.

By incorporating these future improvements, the road defect detection system can evolve into a fully autonomous, intelligent road health monitoring network that significantly enhances road safety and supports smart city initiatives.

Bibliography

[1] R. Kumar and A. Sharma. Iot based pothole detection system using arduino and vibration sensors. *International Journal of Engineering Research Technology*, 9(3):112–116, 2020.

[2] J. Patel and R. Joshi. Pothole detection using accelerometer. *International Journal of Innovative Research in Computer and Communication Engineering*, 6(5):4012–4015, 2018.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[4] A. Singh and S. Mehta. Using gis and mobile data for mapping urban road conditions. *Urban Computing Journal*, 4(2):55–62, 2019.

[5] Liang Zhang, Fang Yang, Nezha Daniel, and Yinhai Wang. Automatic road crack detection using convolutional neural networks. In *Proceedings of the IEEE International Conference on Image Processing*, pages 3708–3712, 2017.

[6] Priya Sharma and Rajiv Gupta. Edge computing for smart city applications: A case study of pothole detection. *IEEE Internet of Things Journal*, 8(6):4700–4710, 2021.

[7] S. Jadhav and M. Bhosale. Sensor fusion techniques for pothole detection: A review. *International Journal of Engineering Science and Technology*, 10(4):24–29, 2018.

[8] OpenStreetMap contributors. Openstreetmap: Open data for navigation and gis. `https://www.openstreetmap.org`, 2023.

[9] A. Deshmukh and S. Kulkarni. A smart city framework for intelligent road condition monitoring. *Smart Systems and Technologies*, 5(1):87–94, 2020.

[10] R. Mishra and P. Singh. Real-time road surface monitoring with cloud and iot integration. *International Journal of Advanced Computer Science and Applications*, 13(2):43–50, 2022.

[11] Arshdeep Bahga and Vijay Madisetti. *Internet of Things: A Hands-On Approach*. Universities Press, 2015.

[12] John Vetelino and Aravind Reghu. *Introduction to Sensors*. CRC Press, 2010.

[13] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[14] John G. Proakis and Dimitris K. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications*. Pearson Education, 4 edition, 2022.

[15] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2016.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[17] Paul A. Longley, Michael F. Goodchild, David J. Maguire, and David W. Rhind. *Geographic Information Systems and Science*. Wiley, 3 edition, 2008.

[18] Yuta Maeda, Yu Sekimoto, Toshiaki Seto, Tohru Kashiyama, and Hiroki Omata. Road damage detection using deep neural networks with smartphone images. *Computer-Aided Civil and Infrastructure Engineering*, 33(12):1127–1141, 2018.

[19] S. Basu, S. Mukherjee, and P. Dutta. Smartphone-based pothole detection and localization using accelerometer, gps and machine learning. *Sensors*, 19(3):591, 2019.

[20] N. Mohan, U. R. Acharya, and A. K. Ray. Vision-based road surface anomaly detection and classification. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1673–1678, 2019.

[21] L. Chen, X. Zhang, and Y. Liu. Machine learning–based pothole severity classification from road surface images. *Transportation Research Part C: Emerging Technologies*, 125:103007, 2021.

[22] H. Wang, J. Li, and F. Zhou. Multi-sensor fusion framework for real-time road condition assessment. *Automatica*, 136:110085, 2022.

[23] P. Li, Q. Xu, and Y. Zhao. Edge-enabled federated learning for privacy-preserving road defect detection. *IEEE Internet of Things Journal*, 10(5):3562–3574, 2023.

[24] T. Nguyen, H. Le, and D. Tran. Crowdsourcing road surface condition monitoring using smartphones and low-cost sensors. In *2020 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pages 45–52, 2020.

[25] C. Zhu, B. Akinci, and P. Fieguth. Automated pavement distress detection from uav imagery using deep learning. *Automation in Construction*, 115:103210, 2020.

[26] R. Singh and P. Kumar. Deep learning based semantic segmentation for road surface anomaly detection. *Journal of Transportation Engineering, Part A: Systems*, 147(8):04021047, 2021.

[27] M. Islam and M. Rahman. Digital image processing techniques for pothole detection: A comprehensive review. *IET Intelligent Transport Systems*, 14(7):566–577, 2020.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude and sincere appreciation to my supervisor, **Dr. P. S. Deshpande** Sir, for offering me the opportunity to work on this project on Designing Metadata Specification Template. His constant support, invaluable guidance, expertise, and encouragement have been instrumental. I am truly fortunate to have the opportunity to work under his mentorship.

I would like to extend my heartfelt thanks to **Dr. M. P. Kurhekar**, Head of Department, for his invaluable support and guidance throughout this journey. His encouragement has been vital in the successful completion of this project.

I would also like to extend my heartfelt thanks to all the teaching and non-teaching faculty members of the Department of Computer Science and Engineering present at Visvesvaraya National Institute of Technology Nagpur for their knowledge sharing and provision of vital resources when necessary.

I would also like to express my gratitude to my family and friends for their unwavering support and encouragement throughout this journey. Their love, motivation, and belief in my abilities have been the driving force behind my accomplishments.

Harshavardhan Ravindra Chatte

Department of Computer Science & Engineering
Visvesvaraya National Institute of Technology, Nagpur

Declaration

I, **Harshavardhan Ravindra Chatte**, hereby declare that this dissertation titled " **IoT-Based Road Defect Detection System** " is carried out by me in the Department of Computer Science & Engineering of Visvesvaraya National Institute of Technology, Nagpur. The work is original and has not been submitted earlier whole or in part for the award of any degree at this or any other Institution.

Harshavardhan Ravindra Chatte

Name and signature

Date: 30-05-2025

Certificate

This to certify that the dissertation titled **"IoT-Based Road Defect Detection System"** is submitted by **Harshavardhan Ravindra Chatte** in partial fulfillment of the requirements for the award of the degree of **Master of Technology in Computer Science & Engineering**, VNIT Nagpur. The work is comprehensive, complete, and fit for final evaluation.

Prof. P. S. Deshpande
Department of Computer Science & Engineering, VNIT Nagpur
प्राध्यापक
Professor
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg
वि.रा.प्रौ.सं. नागपुर-10 (भारत)
VNIT NAGPUR-440010 (India)

Dr. M. P. Kurhekar
Head, Department of Computer Science & Engineering, VNIT Nagpur
Date:
विभाग प्रमुख
Head
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg.
वि.रा.प्रौ.सं. नागपुर-10 (भारत)
VNIT, Nagpur-440010. (India)

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology, Nagpur

Non-Plagiarism Certificate

Certified that M. Tech Dissertation titled **"IoT Based Road Defect Detection System"** submitted by **Harshavardhan Ravindra Chatte**, Enrollment No. **MT23MCS006**, ID No. **30126** has been checked for plagiarism using TURNITIN software and the overlap is found within prescribed limits. A summary of the report is given below.

ORIGINALITY REPORT

9%	7%	5%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

Harshavardhan Ravindra Chatte
Enrollment No: MT23MCS006

Prof. P.S. Deshpande
Professor
Department of Computer Science & Engineering
VNIT, Nagpur

प्राध्यापक
Professor
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg
वि.रा.प्रौ.सं. नागपूर-10 (भारत)
V.N.I.T., NAGPUR-440010 (India)